VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

WEBOVÁ APLIKACE PRO SNADNOU EDITACI KRÁTKÝCH TEXTŮ S OMEZENÝM SLOVNÍKEM

WEB APP FOR SIMPLE EDITING OF SHORT TEXT WITH LIMITED VOCABULARY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **SIMONA POLÁČEKOVÁ**
AUTHOR

VEDOUCÍ PRÁCE — **prof. Ing. ADAM HEROUT, PhD.**
SUPERVISOR

BRNO 2018

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Poláčeková Simona**
Obor: Informační technologie
Téma: **Webová aplikace pro snadnou editaci krátkých textů s omezeným slovníkem**
Web App for Simple Editing of Short Text with Limited Vocabulary
Kategorie: Uživatelská rozhraní

Pokyny:

1. Seznamte se s problematikou vytváření webových aplikací.
2. Prostudujte a zhodnoťte existující nástroje pro editaci krátkých textů s omezeným slovníkem.
3. Navrhněte základní funkčnost řešené aplikace.
4. Implementujte funkční prototyp řešené aplikace.
5. Průběžně testujte prototyp na uživatelích a iterativně vylepšujte / přidávejte funkčnost.
6. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- Steve Krug: Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability, ISBN-13: 978-0321965516
- Susan M. Weinschenk: 100 věcí, které by měl každý designér vědět o lidech, Computer Press, Brno 2012
- Jan Řezáč: Web ostrý jako břitva, Baroque Partners, 2014

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3,
- značné rozpracování bodu 4,
- započetí řešení bodu 5.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Herout Adam, prof. Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
L.S. 612 66 Brno, Božetěchova 2

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Cieľom tejto práce je návrh, implementácia a následne testovanie jednoduchej a užívateľsky prívetivej webovej aplikácie, ktorá používa obmedzený slovník pre kontrolu jazyka. Obmedzeným slovníkom je myslený slovník najčastejšie používaných českých slov. Aplikácia sa dá využiť pre jednoduché písanie textu. Užívateľ si môže vyskúšať, či vie nejakú zložitú myšlienku, vetu alebo text napísať len pomocou najčastejšie používaných slov.

Úvod tejto práce popisuje technológie využité pri vytváraní tejto aplikácie, ich funkčnosť a spôsob použitia. Následne je popísaná implementácia aplikácie a na záver je zhrnutý priebeh testovania aplikácie s užívateľmi a celkové výsledky tejto bakalárskej práce.

Abstract

The aim of this thesis is a draft, an implementation and subsequent testing of a simple and user-friendly web application, which makes use of a limited dictionary to check the language. The limited dictionary is a dictionary of the most commonly used Czech words. The application can be used for simple writing of texts. The user can test if he can write a more complex thought, sentence or text using only the most commonly used Czech words.

The introduction of this thesis describes the technologies used to make this application, their functionality and the way in which they are used. Then there is described implementation of the application. As a conclusion, there is a summary of the testing of the application by users and the overall results of this bachelor thesis.

Kľúčové slová

web, aplikácia, editor, textový editor, jednoduchý, jednoduchosť, top slová, najčastejšie slová

Keywords

web, application, app, editor, text editor, simple, simpleness, top words, most common words

Citácia

POLÁČEKOVÁ, Simona. *Webová aplikace pro snadnou editaci krátkých textů s omezeným slovníkem*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, PhD.

Webová aplikace pro snadnou editaci krátkých textů s omezeným slovníkem

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracovala samostatne pod vedením pána prof. Ing. Adama Herouta, PhD. Uviedla som všetky literárne pramene a publikácie, z ktorých som čerpala.

.....
Simona Poláčeková
16. mája 2018

Poďakovanie

Chcela by som sa poďakovať vedúcemu práce prof. Ing. Adamovi Heroutovi, Ph.D. za odborné vedenie bakalárskej práce a cenné redy pri jej vypracovaní. Taktiež ďakujem svojej rodine a priateľovi za ich podporu a rady.

Obsah

Kapitola 1

Úvod

Táto práca je zameraná na vývoj jednoduchej webovej aplikácie, ktorá bude slúžiť predovšetkým na editáciu textov. Nejde však len o textový editor, pretože pre svoj chod nevyužíva celé pravopisné slovníky českého jazyka, ale len obmedzený slovník najčastejšie používaných českých slov získaných z oficiálnych štatistík[1]. Ide o aplikáciu, ktorá ešte pre český jazyk nebola doposiaľ implementovaná. Snažila som sa o jednoduché a priehľadné riešenie, ktoré bude užívateľský prívetivé, pretože ako sa hovorí, v jednoduchosti je krása. Moje riešenie teda predstavuje pútavú a istým spôsobom zábavnú aplikáciu, ktorá môže vyvolať záujem u rôznych užívateľov.

Hlavnou úlohou aplikácie je poskytovať rozhranie, v ktorom bude možné zadávať užívateľský vstup, a ktorý bude následne kontrolovaný na základe porovnania s výskytom slova vo vybranom rozmedzí najčastejšie používaných slov. Užívateľ teda môže písať ľubovoľný český text, ktorý bude vyhodnotený a slová, ktoré sú mimo určený rozsah, budú viditeľne odlišené. Užívateľ si teda tiež môže zvoliť rozsah slovníka z niekoľkých možností a to od tisíc do desať tisíc najčastejších slov. Taktiež je možné daný text zo stránky uložiť alebo načítať z nejakého súboru. Aplikácia je použiteľná pre rôzne prípady, napríklad ak si niekto chce vyskúšať či vie použiť iba niekoľko základných slov na vyjadrenie nejakej myšlienky, alebo taktiež pre učiteľov (napríklad učiteľov vyučujúcich český jazyk), ktorý by svojim žiakom mohli pomocou tejto aplikácie vysvetliť nejaké náročnejšie pojmy (slová), ktoré by takto boli pochopiteľnejšími a ľahšími na zapamätanie. Taktiež by bola vhodná pre žiakov, ktorý by sa takýmto spôsobom mohli hravou formu učiť český jazyk.

Práca je tvorená pomocou frameworku Symfony[2], ktorý má rôzne užitočné komponenty pre zostavovanie webových aplikácií. V následujúcich kapitolách bude opísaná funkcionalita tohto frameworku a taktiež ako je možné ho využiť pri implementácií v konkrétnych príkladoch. Okrem toho táto bakalárska práca zaoberá spracovaním dát pre vytvorenie slovníka za pomoci programu Majka[3], ktorý slúži na spracovávanie jazyka. Ďalej budú opísané konkrétne postupy pre vytváranie užívateľského rozhrania tejto aplikácie od jeho návrhu, cez implementáciu až po testovanie a zhodnocovanie výsledkov.

[1] Srovnávací frekvenční seznamy – http://ucnk.korpus.cz/srovnani10.php

[2] Symfony, High Performance PHP Framework for Web Development – https://symfony.com/

[3] Free natural language morphology for Czech, Slovak, Polish, Swedish, German, French, Italian, English, Portuguese, Catalan, Welsh, Spanish, Galician, Asturian and Russian – https://nlp.fi.muni.cz/czech-morphology-analyser/

Kapitola 2

Vytváranie webových aplikácií

Pri vytváraní webových aplikácií je v prvom rade potrebné sa zamyslieť, pre koho bude daná aplikácia určená a to najmä preto, že každá aplikácia predstavuje určité užívateľské rozhranie, ktoré by malo byť prispôsobené potrebám užívateľa. Ak však chceme tieto informácie získať, je potrebné s cieľovou skupinou spolupracovať a vždy pomôže opakované testovanie danej aplikácie na užívateľoch. Rovnako dôležité je, aby pri kontakte užívateľa s danou aplikáciou užívateľ nemal problémy s porozumením, ako daná aplikácia funguje.

Čo sa týka konkrétneho vývoja aplikácie, je nutné sa zamerať na vývojové prvky ako sú napríklad technológie, ktoré chceme pri vývoji použiť. Medzi najdôležitejšie technológie pre tvorbu stránky patria programovacie jazyky ako je HTML, JavaScript, CSS a iné. Samotná webová stránka sa dá vytvoriť pomocou základného jazyka HTML, no v dnešnej dobe by sa dalo povedať, že už neexistuje takmer žiadna webová stránka či aplikácia, ktorá by nebola tvorená taktiež pomocou programovacích jazykov JavaScript či CSS. Existuje však už aj mnoho ďalších technológií, ktoré slúžia pre vytváranie webu ako napríklad jQuery a Bootstrap, ktoré som taktiež využila pri tvorbe tejto aplikácie pre editovanie textu s obmedzeným slovníkom. Okrem toho sa dajú použiť taktiež rôzne frameworky, ako je napríklad Symfony, ktoré prácu s týmito technológiami zjednodušujú.

Táto kapitola opisuje, ako pracovať s frameworkom Symfony pri vytváraní webových aplikácií. Taktiež sú v tejto kapitole opísané technológie, ktoré som používala pri vývoji webovej aplikácie a to najmä spôsob, akým fungujú a ako sú navzájom prepojené. Veľké množstvo aplikácií si okrem toho žiada spracovanie vonkajších dát, ktoré budú následne používané v danej aplikácii. Príkladom takýchto dát môže byť získanie a spracovanie slovníka, čo bolo potrebné v mojom prípade. Na to som využila prgoramovací jazyk Java. Základné informácie ohľadom tohto jazyka sú taktiež zhrnuté v tejto kapitole a to v sekcii 2.4.

2.1 Vytváranie webových aplikácií vo frameworku Symfony

Vo všeobecnosti je framework nástroj, ktorý zjednodušuje vývoj aplikácií automatizovaním mnohých vzorov objektového programovania. Framework taktiež vytvára nad kódom štruktúru, čo užívateľa núti písať lepší, čitateľnejší a hlavne lepšie udržiavaný kód. Okrem toho sa vďaka frameworku ľahšie programuje, pretože zjednodušuje komplexné riešenia do niekoľkých jednoduchých príkazov.

Framework Symfony bol navrhnutý pre zjednodušenie vývoju webových aplikácií prostredníctvom niekoľkých kľúčových znakov. Od väčšiny ostatných frameworkov sa líši predovšetkým tým, že definuje návrhový vzor Model-View-Controller (MVC), zatiaľ čo mnoho

iných frameworkov sa snaží len spĺňať jeho pravidlá [7]. Symfony obsahuje početné množstvo nástrojov a tried, vytvorených pre skrátenie času na vývoj komplexných webových aplikácií. Dokonca taktiež automatizuje bežné úlohy, aby sa vývojár mohol celkovo sústrediť len na vývoj svojej aplikácie.

Tetno framework bol vytvorený pomocou jazyka PHP. Bol otestovaný na mnohých projektoch a využíva sa pre vývoj biznisových webových stránok s vysokými požiadavkami [9].

Pre vytváranie mojej webovej aplikácie som použila konkrétne framework Symfony 4, ktorý predstavuje riešenie s menším množstvom konceptov Symfony a viacero štandardných metód. Táto nová verzia vylepšuje pôvodné riešenia tak, aby v nej vývojár vedel rýchlejšie prosperovať, aby sa naučil rýchlejšie pracovať v danom frameworku a taktiež je táto nová verzia jednoduchšia na konfiguráciu, inštaláciu, nasadenie a ľahšia na osvojenie[1]. Najmä pre tieto charakteristiky som si vybrala túto verziu frameworku.

2.2 Symfony a jeho hlavné znaky

Symfony bolo vytvorené predovšetkým tak, aby spĺňalo nasledujúce požiadavky:

- Jednoduché na inštaláciu a konfiguráciu na väčšine platformách (zaručená funkčnosť na štandardných *nix a Windows platformách).
- Nezávislé na databázových zariadeniach.
- Vo väčšine prípadov jednoduché na použitie, no taktiež dostatočne flexibilné na riešenie komplexných úloh.
- Založené na predpoklade, že je konvencia prednejšia pred konfiguráciou – je potrebné aby vývojár nakonfiguroval len to, čo nie je bežne zaužívané.
- V súlade s väčinou najlepších webových postupov a návrhovými vzormi.
- Prispôsobené existujúcim informačným technológiam (IT) a ich architekturám, dostatočne stabilné pre dlhodobé projekty.
- Čitateľný kód s komentármi pomocou phpDocumentor, pre jednoduché udržiavanie.
- Jednoducho rozšíriteľné, povoľujúce integráciu s ostatnými knižnicami [9].

Framework Symfony obsahuje veľké množstvo komponentov, ktoré predstavujú súbor samostatných a znovu použiteľných PHP knižníc. Stavajú sa základom, na ktorom sú postavené najlepšie PHP aplikácie. Tieto komponenty sa dajú využiť v akejkoľvek aplikácii nezávislé na tomto frameworku. Medzi hlavné komponenty, ktoré by som chcela spomenúť patria:

- Config (od slova configuration – prispôsobenie) – pomáha vyhľadávať, načítať, kombinovať, automaticky dopĺňať a potvrdzovať nakonfigurované hodnoty.
- Routing (smerovanie) – mapuje požiadavky HTTP na konfiguračné premenné.
- Templating (šablónovanie) – poskytuje prostriedky potrebné na vytvorenie akéhokoľvek šablónového systému.

[1] Symfony 4, a high-performance PHP framework and a set of components. – https://symfony.com/4

- Filesystem (súborový systém) – poskytuje základné služby pre súborové systémy.
- Lock (zámok) – vytvára a spravuje zámky, mechanizmy, ktoré poskytujú výlučný prístup k zdieľaným zdrojom.
- Messenger – pomáha aplikáciam medzi sebou príjmať a odosielať správy.
- Console (ovládací panel) – Uľahčuje vytváranie prívetivých a testovateľných rozhraní pre príkazový riadok.

Okrem týchto súčastí, ktoré som pri vytváraní aplikácie použila, respektíve by sa dali použiť pri jej ďalšom vývoji, má Symfony aj mnoho ďalších užitočných komponentov[2].

Ďalšie vlastnosti, ktorými Symfony disponuje, sú napríklad rozšíriteľná objektovo orientovaná architektúra a pomocné vývojárske prostriedky, ktoré umožňujú vývoj aplikácií. Znamená to, že každá časť frameworku môže byť doplnená, prispôsobená, nahradená či odstránená. Taktiež je tento framework prispôsobený na prácu s rôznymi dátovými štruktúrami či technológiami. Je postavený na moderných praktikách a štandardoch. Na rozdiel od ostatných frameworkov nemá obmedzenia, ako by mala vyvíjaná aplikácia pracovať. Práve naopak je možné aplikáciu prispôsobiť vlastným potrebám. Pri vytváraní rozhrania pre užívateľský náhľad, Symfony používa šablónovací systém Twig, ktorý rapídne urýchľuje túto prácu. Hlavné verzie tohto frameworku sú väčšinou podporované tri až štyri roky[3].

2.2.1 Composer

Composer je nástroj pre spravovanie závislostí v jazyku PHP. Umožňuje deklaráciu knižníc, na ktorých závisí vyvíjaný projekt a práve pomocou Composera sú dané knižnice spravované (dokáže ich napríklad inštalovať či aktualizovať).

Tento nástroj však neobsahuje nové myšlienky, je inšpirovaný na základe iných riešení (nástroj `npm` pre *Node.js* a `bundler` pre *Ruby*).

V situácii, kedy existuje projekt, ktorý je závislý na niekoľkých knižniciach a niektoré tieto knižnice taktiež závisia na iných knižniciach, pracuje Composer následovne:

- povoľuje deklaráciu knižníc, na ktorých daný projekt závisí,
- zistí, ktoré verzie, ktorých balíkov môžu a potrebujú byť nainštalované, a následne ich nainštaluje (stiahne ich do daného projektu).

Composer takto uľahčuje riešenie závislostí, užívateľ nemusí poznať závislosti medzi knižnicami, tie sú automaticky pridané tam, kde je to potrebné.

Tento nástroj je multi-platformný, čo znamená, že je podporovaný na rôznych platformách (napríklad Windows, či Linuxové distribúcie)[4].

Composer je teda správca balíkov, používaných modernými PHP aplikáciami. Preto pri vývojí mojej práce vo frameworku Symfony pomáhal spravovať závislosti a taktiež pomáhal s inštaláciou komponentov Symfony v mojom PHP projekte[5].

[2] Symfony Components – https://symfony.com/components

[3] Introducing Symfony | Unleashed Technologies – https://www.unleashed-technologies.com/blog/2017/01/31/introducing-symfony

[4] Introduction - Composer – https://getcomposer.org/doc/00-intro.md#downloading-the-composer-executable

[5] Installing Composer (Symfony Docs) – https://symfony.com/doc/current/setup/composer.html

2.3 Technológie pre vytváranie webových aplikácií

Technológie pre vytváranie webových stránok či aplikácií sa zvyčajne rozdeľujú na tie, ktoré vytvárajú obsah danej stránky – jazyky HTML, XHTML, XML a ďalšie, tie ktoré pridávajú stránke nejaké štýlové predpisy – jazyk CSS, a technológie, ktoré s danou stránkou pracujú prostredníctvom rôznych interakčných funkcií – jazyk JavaScript a iné. Spomínané technológie sú navzájom prepojiteľné tak, aby spolu vytvárali rozumné riešenie pre kompletnú realizáciu webových stránok.

V tejto sekcii popisujem funkcionalitu technológií, ktoré som použila pri vytváraní mojej webovej aplikácie.

2.3.1 HTML

Hypertext Markup Language, v skratke HTML, je najviac rozšírený jazyk pre vytváranie webových stránok. Ako aj názov napovedá, jedná sa o značkovací jazyk (markup language). So značkovaním sa stretávame takmer každodenne, jedná sa o dodanie určitej vlastnosti dokumentu tak, aby táto vlastnosť pridávala dokumentu nejaký špecifický význam. V bežnom živote je to napríklad používanie zvýrazňovača pre odlíšenie nejakých častí textu. Vo webovom kontexte je značkovaním myslené predávanie určitej štruktúry dokumentu a samotné označkovanie predstavuje, ktorá časť dokumentu je hlavičkou, ktoré časti predstavujú paragraf, tabuľku, telo dokumentu a podobne. Tento jazyk slúži na to, aby sa daný dokument vhodne zobrazil v nejakom webovom prehliadači [2].

Symfony – šablónovací systém Twig

Symfony používa pre vytváranie dokumentov šablónovací systém Twig vyvinutý pre PHP. Hlavnými znakmi tohto šablónovacieho systému sú[6]:

- Rýchlosť – Twig zostavuje šablóny spôsobom jednoduchého optimalizovaného PHP kódu. Pri porovnaní s bežným PHP kódom je horná hranica náročnosti zredukovaná na minimum.
- Bezpečnosť – Twig má svoju modifikáciu s názvom `sandbox` pre spúšťanie nedôveryhodného šablónového kódu. To dovoľuje, aby bol Twig používaný ako šablónovací jazyk pre aplikácie, kde užívatelia môžu modifikovať dizajn danej šablóny.
- Flexibilita – Twig je založený na prispôsobivom lexikálnom a syntaktickom analyzátore. Vďaka tomu si môžu jeho užívatelia (vývojari) definovať ich vlastné označenia (ďalej v texte označené ako tagy) a filtre, či dokonca vytvoriť vlastný DSL (domain-specific language).

Šablóny v základe predstavujú len textové súbory obsahujúce premenné a výrazy, ktoré sú nahradené ich hodnotami po vyhodnotení danej šablóny. Dôležitou súčasťou šablonového súboru sú taktiež tagy, pretože kontrolujú logiku samotnej šablóny. Šablóna môže obsahovať kód html, ktorý je rozšírený o použitie danej šablónovacej logiky. Systém Twig môže byť taktiež rozšírený o dodatočné tagy, filtre, testy, operátory, globálne premenné a funkcie. Viac informácií o rozšírení Twigu sa nachádza v oficiálnej dokumentácii[7]. Twig má dva primárne jazykové prvky:

[6] Home - Twig - The flexible, fast, and secure PHP template engine – https://twig.symfony.com/

[7] Extending Twig - Documentation - Twig - The flexible, fast, and secure PHP template engine – https://twig.symfony.com/doc/2.x/advanced.html

- `{{ }}` – zobrazí výsledok vyhodnoteného výrazu,
- `{% %}` – vykoná daný príkaz.

Príklad jednoduchej šablóny vytvorenej pomocou Twig:

```
<!DOCTYPE html>
<html>
    <head>
        <title>All About Cookies</title>
    </head>
    <body>
        My name is {{ name }} and I love cookies.
        My favorite flavors of cookies are:
        <ul>
        {% for cookie in cookies %}
            <li>{{ cookie.flavor }}</li>
        {% endfor %}
        </ul>
        <h1>Cookies are the best!</h1>
    </body>
</html>
```

V tomto príklade je ukážka bežného použitia jazyka HTML pre vytvorenie štandardnej webovej stánky. Čo však nie je štandardom, je práve využitie syntaxe šablónovacieho systému Twig pre prezentáciu mena autora stránky (`{{ name }}`) a taktiež vytvorenie dynamického zoznamu druhov položiek (`{% for cookie in cookies %} ... {% endfor %}`). Širší popis s touto, ale aj s inými ukážkami, sa nachádza online na webovej stránke[8].

2.3.2 CSS

Kaskádové štýly CSS (skratka z anglického cascading style sheets) je programovací jazyk, pomocou ktorého je možné pridať (HTML) dokumentom určitý štýl. CSS používa pravidlá, pomocou ktorých definuje predpisy štýlu pre daný dokument, a teda definuje, ako sa bude dokument zobrazovať. Tieto pravidlá je zvyčajne lepšie písať do osobitného dokumentu než do konkretného súboru s obsahom na zobrazenie, aby sa zachovala štruktúra pôvodného dokumentu.

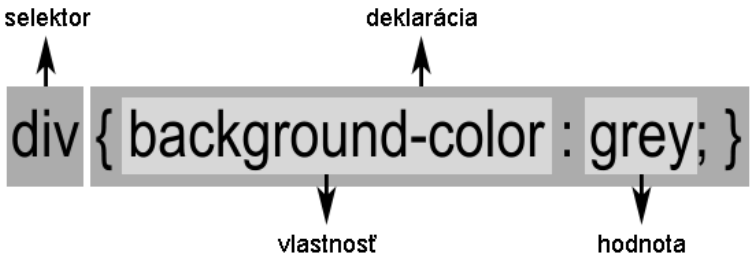
CSS pravidlo sa skladá z dvoch častí:

- Selektor – definuje, na ktorý element, respektíve elementy bude pravidlo aplikované. (Pri viacerých elementoch sa dá použiť napríklad zoznam selektorov, oddelených čiarkami.)
- Deklarácia – nastavuje štýl, ktorý ma byť na elementy aplikovaný.

Samotná deklarácia sa takiež delí na dve časti, ktoré sú rozdelené dvojbodkou a to:

- Vlastnosť – jedná sa o vlastnosť elementu (elementov), ktoré chce autor nakonfigurovať podľa seba. Príkladom takejto vlastnosti je farba pozadia daného elementu (`background-color`).

[8] Twig Primer | Grav Documentation – https://learn.getgrav.org/themes/twig-primer



Obr. 2.1: CSS pravidlo na tomto obrázku sa vzťahuje na elementy `<div>`. Jedná sa o typový selektor (`div`). Pravidlo vyjadruje, že sa farba pozadia (`background-color`) daných elementov zmení na hodnotu sivej farby (`grey`).

- Hodnota – špecifikuje danú vlastnosť. Napríklad určenie zelenej farby pre pozadie (alebo pre akúkoľvek inú vlastnosť elementu, ktorá určuje farbu – `green`).

Konkrétny príklad použitia CSS je zobrazený aj na obrázku 2.1.

Pre špecifikáciu CSS pravidla je rovnako dôležitý aj selektor. Medzi najdôležitejšie selektory patria predovšetkým:

- Typový selektor – špecifikuje pravidlá pre element podľa jeho názvu ako napríklad selektor `p` pre paragraf, selektor `body` pre telo dokumentu a podobne.
- Triedny selektor – špecifikuje pravidlá pre všetky elementy, ktoré majú určený atribút `class` v HTML dokumente. Viaceré elementy môžu mať tu istú triedu (`class`) a zároveň každý element môže mať priradené viaceré triedy, ktoré sú oddelené medzerou. Tento selektor sa označuje pomocou bodky a názvu danej triedy (napríklad pre element, ktorý má určenú triedu s názvom `stred`, bude označený ako `.stred`).
- Selektor prostredníctvom identifikátora – funguje na podobnom princípe ako triedny selektor, ale vzhľadom k tomu, že by mal byť atribút identifikátor (`id`) unikátny pre každý element, aj toto pravidlo by sa malo vzťahovať len na jeden element. Aby užívateľ označil tento selektor, potrebuje ho zapísať pomocou mriežky a daného identifikátora (príkladom je `#hlavicka`).

Okrem týchto najdôležitejších selektorov existujú aj mnohé ďalšie modifikácie, ktoré pomáhajú charakterizovať presný výber elementov, na ktorý má vývojár v úmysle aplikovať pravidlo CSS. Sú to napríklad univerzálny selektor (`*`), selektor dieťaťa (`div > a` – vyberá všetky elementy `<a>`, ktoré sú priamymi potomkami elementov `<div>`), zostupný selektor (`div a` – vyberá všetky elementy `<a>`, ktoré sú potomkami elementov `<div>`, no nemusia byť len priamymi potomkami, jedná sa aj o všetky vnorené elementy), a ďalšie.

Ak má jeden element menenú jeho jednu vlastnosť v niekoľkých pravidlách, výsledná hodnota bude zvolená podľa dôležitosti pravidiel. Pravidlo, ktoré je špecifikované identifikátorom má najvyššiu hodnotu, následne je to triedny selektor, typový selektor a poradie daného pravidla v dokumente, kde vyhráva posledné pravidlo [2].

2.3.3 Bootstrap

Bootstrap predstavuje jednu z najčastejšie používaných knižníc pre vytváranie grafického rozhrania pre webové aplikácie. Jedná sa o voľne dostupné riešenie pre vývoj s jazykmi

HTML, CSS a JavaScript. Pomocou tejto knižnice sa dá veľmi jednoducho vybudovať responzívnu webovú aplikáciu a teda aplikáciu, ktorej grafický dizajn sa vie prispôsobiť podľa toho, na akom zariadení má byť zobrazená.

Tento framework sa veľmi jednoducho používa a je tvorený z predpripravených štýlov, prostriedkov pre rozostavenie elementov a interaktívnych súčastí, čo vyvojárom umožňuje vytvárať webové stránky a aplikácie, ktoré sú vizuálne príjemné a bohaté na funkcionalitu. Bootstrap sa dá aplikovať na široké množstvo značkovacích štruktúr. Za interaktívne súčasti sa považujú napríklad modálne dialógy, roztváracie ponuky či rôzne iné pomôcky.

2.3.4 JavaScript

Programovací jazyk JavaScript (taktiež nazývaný ako *ECMAScript*) je objektovo orientovaný jazyk pre webové prehliadače. Umožňuje vyvojárom na webových stránkach robiť nasledujúce:

- Čítať elementy z dokumentov, či vytvárať a zapisovať nové elementy do dokumentov.
- Narábať s textom, alebo ho posúvať.
- Vytvárať vyskakovacie okná.
- Uskutočňovať matematické výpočty nad dátami.
- Reagovať na udalosti, ako napríklad prejdenie myšou cez nejaký objekt, stlačenie klávesy či myši užívateľom a mnoho ďalších.
- Získať aktuálny čas z užívateľovho počítača, alebo posledný čas, kedy bol dokument modifikovaný.
- Určiť veľkosť užívateľovej obrazovky, verziu prehliadača či rozlíšenia obrazovky.
- Uskutočňovať akcie založené na podmienkach, ako napríklad upozorniť užívateľov ak zadajú nesprávne informácie do formulára, alebo keď stlačia určité tlačidlo.

Hlavná myšlienka, na ktorej bol založený vznik tohto programovacieho jazyka, bola, aby boli do webových dokumentov pridané interaktívne prvky, ktoré do tej doby boli len statické.

2.3.5 jQuery

Knižnica jQuery uľahčuje programátorom prácu pri vytváraní webových stránok, pretože pomáha vývojárom s bežnými úlohami a zjednodušuje tie náročné. Jedná sa o knižnicu programovacieho jazyka JavaScript. Táto knižnica sa stalo obľúbenou, pretože poskytuje relatívne jednoduché riešenia pre širokú škálu úloh.

Knižinca jQuery sa predovštkým používa na webové skriptovanie. Základné funkcie tejto knižnice sú nápomocné pri nasledujúcich úlohách:

- Prístup k jednotlivým elementom v dokumente. Funguje to na základe jQuery selektorov, ktoré sú založené na koncepte CSS selektorov.
- Upravovanie vzhľadu webových stránok. jQuery dokáže meniť triedy a jednotlivé vlastnosti použité na rôzne časti dokumentu a to aj po tom, čo je webová stránka zobrazená.

- Zmeny obsahu dokumentu. Pomocou tejto knižnice je možné meniť obsah dokumentu, napríklad meniť text, vkladať či meniť obrázky alebo upravovať obsah celej štruktúry HTML dokumentu. Všetky tieto zmeny sú vykonávané pomocou jednoduchého rozhrania API (Application Programming Interface).
- Reagovanie na uživateľové činnosti, ako napríklad kliknutie na tlačidlo myši, či klávesnice a mnoho ďalších. jQuery taktiež odstráňuje rozdiely v rozhraní API pri obsluhe daných udalostí na rôznych prehliadačoch.
- Animovanie zmien v dokumente. Existuje totiž mnoho efektov, ktoré táto knižnica ponúka, napríklad posuvanie dokumentu bez pomoci posuvnej lišty od uživateľa, alebo presvítanie či tvorbu nových vizuálnych efektov.
- Načítavanie dát zo serveru bez obnovovania stránky. Dá sa tak učiniť pomocou technológie Ajax (Asynchrnous JavaScript and XML).
- Zjednodušovanie bežných činností pri programovaní v jazyku JavaScript. Okrem nových funkcií taktiež rozširuje základné konštrukcie pre JavaScript.

Táto knižnica využíva koncepty jazyka CSS pre vyjadrenie štruktúry dokumentu a preto je jednoduchá na použitie. Konkrétne sú to selektory jazyka CSS, pomocou ktorých sa daná štruktúra vyjadruje. Okrem toho je taktiež rozšíriteľná vzhľadom k tomu, že umožnuje tvorbu zásuvných modulov. Zaujímavosťou je, že pomocou jQuery je možné reťazenie príkazov. Takto sa dá vykonať viacero akcií len v rámci jedného riadku [1].

2.4 Spracovanie dát – programovací jazyk Java

Tento programovací jazyk som pri vytváraní webovej aplikácie použila na spracovanie dát, ktoré daná aplikácia využívala pre svoj chod. Konkrétne som pomocou Javy vytvorila referenčný slovník, na základe ktorého sa porovnáva užívateľov vstup. V tejto sekcii preto opisujem niektoré funkcionality programovacieho jazyka Java, ktoré som pri svojej práci využila.

Programovací jazyk Java je objektovo orientovaný jazyk založený na triedach. Je navrhnutý tak, aby bol dostatočne jednoduchý na to, že v ňom veľké množstvo programátorov dosiahne schopnosť plynulého písania kódu. Je v úmysle, aby sa jednalo o jazyk produkcie a nie o vedecký jazyk, preto sa dizajn tohto jazyka vyhýbal zahrňovaniu nových a neotestovaných prvkov.

Java je silno typovaný jazyk. Táto špecifikácia rozlišuje medzi chybami pri kompilácií, ktoré môžu a musia byť detekované počas kompilácie a tými, ktoré sa vyskytujú pri behu programu. Programy v tomto jazyku sú písané v znakovej sade Unicode.

Jedná sa o relatívne vysoko úrovňový jazyk, ktorý obsahuje automatické spravovanie pamäte, typicky použitím *garbage collector* na vyhnutie sa bezpečnostným probémom s explicitným uvoľňovaním pamäte. Jazyk neobsahuje žiadne nebezpečné konštrukcie, ako napríklad pristupovanie k poľu bez kontroly indexu, pretože takéto konštrukcie by mohli spôsobiť nežiadané správanie programu.

Vzhľadom k tomu, že má tento jazyk široké rozmädzie použitia, opíšem len základné konštrukcie a funkcie, ktoré som využila pre svoju bakalársku prácu. Pri písaní tejto sekcie som sa inšpirovala z kníh o tomto programovacom jazyku ([4], [8], [5] a [3]).

2.4.1 Triedne typy

Medzi triedne typy programovacieho jazyka Java patria triedy, rozhrania, polia a metódy. Tieto triedne typy sú buď deklarované v type, alebo zdedené, pretože sú dostupné z nadradenej triedy alebo nadradeného rozhrania, ktoré nie sú súkromné ani skryté či prepísané.

Členmi triednych typov sú všetky z nasledujúcich:

- Členy zdedené z priamej nadradenej triedy, ak ju však majú (trieda `Object` nemá žiadnu nadradenú triedu).
- Členy zdedené z akéhokoľvek priamo nadradeného rozhrania.
- Členy deklarované v tele triedy.

Konštruktory nepredstavujú členov triedneho typu.

Java umožňuje preťažovanie metód, čo znamená, že pri definícii metód v triede, ktorá zdedí metódu s rovnakým názvom a počtom argumentov, bude daná metóda prepísaná na novo definovanú a teda bude metóda preťažená. Toto však nie je možné, ak sa pred metódu pridá identifikátor `final`[9].

2.4.2 Lambda výrazy

Lambda výraz je nepomenovaný úsek kódu (alebo nepomenovaná funkcia) so zoznamom formálnych parametrov a telom. Telo lambda výrazu môže byť vyjadrené v bloku alebo pomocou výrazu. Zoznam parametrov je rovanko ako u metód uzavrený v okrúhlych zátvorkách. Šípka (`->`) sa používa pre oddelenie zoznamu parametrov a tela. Ak je telo napísané pomocou bloku kódu, je daný blok uzavretý v zložených zátvorkách. Termín „lambda“ v „lambda výraz“ má pôvod v Lambda kalkule, ktorý používa písmeno gréckej abecedy (λ) pre naznačenie abstrakcie funkcie. Príklad využitia lambda výrazu:

```
(int x, int y) -> {
    int max = x > y ? x : y;
    return max;
}
```

Daný lambda výraz má dva parametre (`x` a `y`) a ako výsledok vráti prameter, ktorý predstavuje väčšie číslo.

Lambda výraz je odlišný od metód a to týmito znakmi:

- Lambda výraz nemá meno.
- Lambda výraz nemá návratový typ. Je odvodený kompilátorom z kontextu použitia a z obsahu tela daného výrazu.
- Lambda výraz nemá klauzulu `throws`, ktorá sa používa pre vyvolanie nejakej výnimky.
- Lambda výraz nemôže deklarovať typové parametre, pretože lambda výraz nemôže byť generický.

[9] Java - preťažovanie metód - kiwiki – http://www.kiwiki.info/index.php/Java_-_pre%C5%A5a%C5%BEovanie_met%C3%B3d

Hlavným cieľom lambda výrazov bolo zanechať ich syntax stručnú a ponechať odvodenie detajlov na prekladač. Pre deklarovanie lambda výrazov existuje tiež skrátená syntax a to vypustením typov parametrov, či vynechaním zátvoriek pri jednom parameteri, alebo vynechaním zložených zátvoriek pre telo, ktoré obsahuje len výraz, ktorý predstavuje výsledok lambda výrazu. Pri vypustení typov parametrov je možné toto skrátenie vykonať len vtedy, ak sa vynechajú typy všetkých parametrov výrazu. V lambda výrazoch je možné použiť vyjadrenia ako `break`, `continue`, `return` a `throw`. Vyjadrenia `break` a `continue` špecifikujú lokálny návrat do tela lambda výrazu, zatiaľ čo `return` a `throw` ukončujú telo lambda výrazu.

Poly výraz je výraz, ktorého typ závisí od kontextu jeho použitia. Lambda výraz je vždy *poly* výrazom. Nemôže sa použiť samostatne. Jeho typ je odvodený prekladačom na základe kontextu. Lambda výraz môže byť použitý v priraďovaniach, pri invokácii metód a návratoch.

2.4.3 Procesy

Java program môže komunikovať s externými procesmi prostredníctvom streamov (stream – prenášanie, vysielanie dát) rovnakým spôsobom, ako pri komunikácii so serverom bežiacom na nejakom inom počítači na sieti. Použitie triedy `java.lang.Process`, ktorá interpretovi predstavuje externe bežiace procesy operačného systému, je vždy závislá na platforme, v ktorej beží a je zriedkavo prenosná.

Pred piatou verziou Javy (*Java 5*) bola metóda `Runtime.exec`, ktorá je súčasťou jedinou metódou, pomocou ktorej bolo možné spustiť vonkajší príkaz priamo z aplikácie napísanej v tomto programovacom jazyku. Bolo to možné prostredníctvom procesov z triedy `java.lang.Process`. *Java 5* však do svojej funkcionality oproti pôvodným riešeniam zaradila novú triedu – `ProcessBuilder`, ktorá umožňuje väčšiu kontrolu nad vygenerovanými procesmi v operačnom systéme. Umožňuje kontrolu premenných prostredia prostredníctvom rozhrania `Map` a je tak jednoduché nastaviť pracovný priečinok. Taktiež má možnosť automaticky presmerovávať spúšťaný štandardný chybový stream procesov na štandardný výstupný stream. Vďaka tomu je oveľa jednoduchšie čítať celý výstup daného procesu (`Process`).

Kapitola 3

Návrh textového editora s obmedzeným slovníkom

V prvej časti návrhu tejto webovej aplikácie som sa zaoberala rozborom zadania. Mojou úlohou je vytvorenie uživateľského rozhrania, pre ktorého použiteľnosť sú dôležité hlavne tieto štyri kroky [10]:

- Objavovanie – pochopenie zadania od klienta (v mojom prípade vedúceho práce) a naplánovanie samotného návrhu
- Uživateľský výskum – pochopenie potrieb užívateľov danej aplikácie
- Návrh aplikácie – zhodnotenie získaných informácií a vytvorenie systematického riešenia
- Evaluácia – testovanie správnosti a použiteľnosti riešenia, vylepšovanie riešenia na základe získaných výsledkov (táto fáza môže prebiehať v niekoľkých iteráciách)

V tejto kapitole popisujem v jednotlivých sekciách ako som analogicky postupovala pri návrhu a teda zhodnotením cieľovej skupiny, vyhľadaním existujúcich riešení, ktorými som sa následne inšpirovala pri návrhu grafického rozhrania aplikácie, potom som sa venovala spracovávaniu dát a postupne uvažovala o hlavných funkciách, ktoré by aplikácia mala ovládať.

3.1 Cieľová skupina

Zhodnotenie cieľovej skupiny je pri vytváraní či už webovej, alebo akejkoľvek inej aplikácie dôležitým krokom. Význam tohto kroku spočíva predovšetkým v tom, aby výsledok aplikácie nebol nezmyslom len preto, že nebude predstavovať výsledok, ktorý užívatelia potrebujú. Je nutné zistiť motiváciu a očakávania užívateľov pre používanie aplikácie. Prečo a kedy ju vôbec potrebujú, ako o nej premýšľajú a ako s ňou narábajú.

Aplikácia opísaná v tejto bakalárskej práci je určená pre akúkoľvek osobu, ktorá by si chcela vyskúšať skladať zmysluplné vety len za pomoci niekoľkých najjednoduchších slov českého jazyka. Aplikácia nie je určená pre špecifickú vekovú kategóriu, samozrejmosťou je, že daný človek musí ovládať písanie na klávesnici či už mobilných telefónov alebo počítačov a mať prístup na internet.

Možnými príkladmi užívateľov sú napríklad deti základnej školy, ktoré sa chcú naučiť nejaké zložité pojmy a túto aplikáciu využijú pre zjednodušenie definície, ktorá následne

bude ľahšie zapamätateľná. Taktiež môže byť vhodná pre učiteľov, ktorý môžu podobne vysvetliť pojmy a definície svojím žiakom jednoduchšou formou pomocou tejto aplikácie. Okrem toho by sa dala využiť ako zábavná vyuková aplikácia českého jazyka. Následne sa o túto aplikáciu môžu zaujímať rôzny užívatelia, ktorý majú obľubu v jednoduchosti a môžu si takto skontrolovať náročnosť svojho prejavu. Taktiež by bola vhodná pri písaní odborného textu tak, aby mu porozumeli aj bežný ľudia. Mohli by ju využívať napríklad aj lekári v svojich lekárskych správach, ktoré píšu pre svojich pacientov, aby danej spáve viac rozumeli.

3.2 Existujúce riešenia

Pri vyhľadávaní riešení, ktoré by aspoň čiastočne zodpovedali môjmu zadaniu som narazila v prvom rade na aplikáciu s názvom „The Up-Goer Five Text Editor[1]“, ktorá je založená na rovnakom princípe. Ide totiž o aplikáciu, ktorá využíva tisíc najzakladnejších slov a kontroluje užívateľský vstup, ktorý porovnáva s daným slovníkom. Ide však o riešenie, ktoré doposiaľ nebolo vyvinuté pre český jazyk. (Existuje pre anglický a španielsky jazyk.) Taktiež som našla iné riešenia, ktoré ale neboli vyvinuté ako webové, ale len ako desktopové aplikácie.

„The Up-Goer Five Text Editor“ bol inšpirovaný obrázkom 3.1, kde je zobrazený komiks, v ktorom je vysvetlená stavba kozmickej lode tak, aby rozumeli aj tí najmenší. Príklady ako funguje toto riešenie sa nachádzajú na obrázkoch 3.2 a 3.3. Na obrázkoch je vidieť, že sa jedná o veľmi jednoduché riešenie. Na základe tohto riešenia je inšpirovaná moja práca.

3.3 Vytváranie slovníka

Vytváranie slovníka bolo jednou z hlavných úloh. Vzhľadom k tomu, že čeština je rozsiahly jazyk, ktorý nepoužíva len základné tvary slov, ale každé slovo má aj svoje skloňovanie alebo časovanie, bolo nutné tomu tento slovník prispôsobiť.

Začala som vyhľadaním dát, ktoré budú predstavovať najčastejšie používané české slová. Tie som našla na stánkach Ústavu Českého národného korpusu[2]. Tieto frekvenčné zoznamy však neobsahujú slová len v základných tvaroch. Aby bol výsledný slovník ľahšie spracovateľný, musí mať v sebe dáta vo formáte, s ktorým sa bude dať ľahšie pracovať. Preto som si navrhla tvar slovníka spôsobom, že bude obsahovať na jednom riadku slovo so všetkými jeho tvarmi. Na to som však zo stiahnutých frekvenčných zoznamov musela získať z každého slova v prvom rade jeho základný tvar (lemmu) a následne všetky jeho tvary. S touto úlohou mi pomohol voľne dostupný program Majka[3].

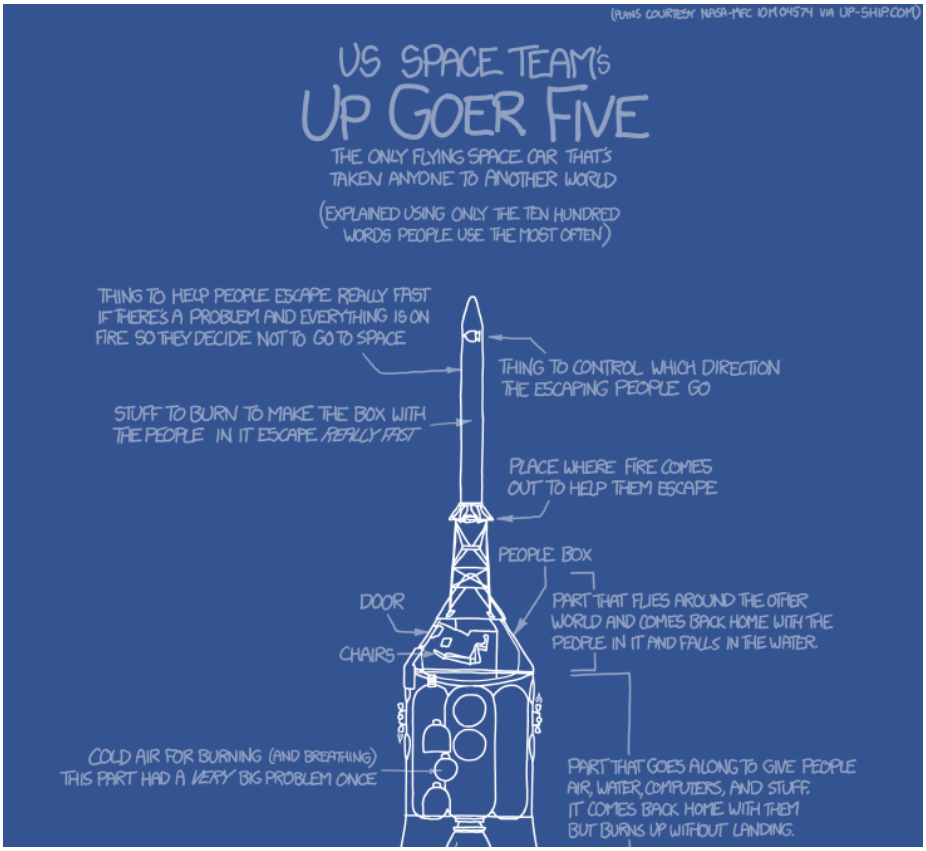
3.3.1 Majka – program na spracovávanie jazyka

Program Majka je voľne dostupný program, ktorý pracuje ako morfologický analyzátor jazyka. Využívala som ho na operačnom systéme Linux, kde sa spúšťa prostredníctvom terminálu. Pri spúšťaní má povinný parameter a to vstupný súbor – morfologickú databázu,

[1] The Up-Goer Five Text Editor – http://splasho.com/upgoer5/

[2] Srovnávací frekvenční seznamy – http://ucnk.korpus.cz/srovnani10.php

[3] Free natural language morphology for Czech, Slovak, Polish, Swedish, German, French, Italian, English, Portuguese, Catalan, Welsh, Spanish, Galician, Asturian and Russian – https://nlp.fi.muni.cz/czech-morphology-analyser/



Obr. 3.1: Komiks UP GOER FIVE – https://imgs.xkcd.com/comics/up_goer_five.png, z ktorého bolo inšpiroavné vytvorenie aplikácie „The Up-Goer Five Text Editor“.

ktorá bola dostupná z rovnakej stránky ako binárne súbory tohto programu, spomenuté v predchádzajúcej sekcii.

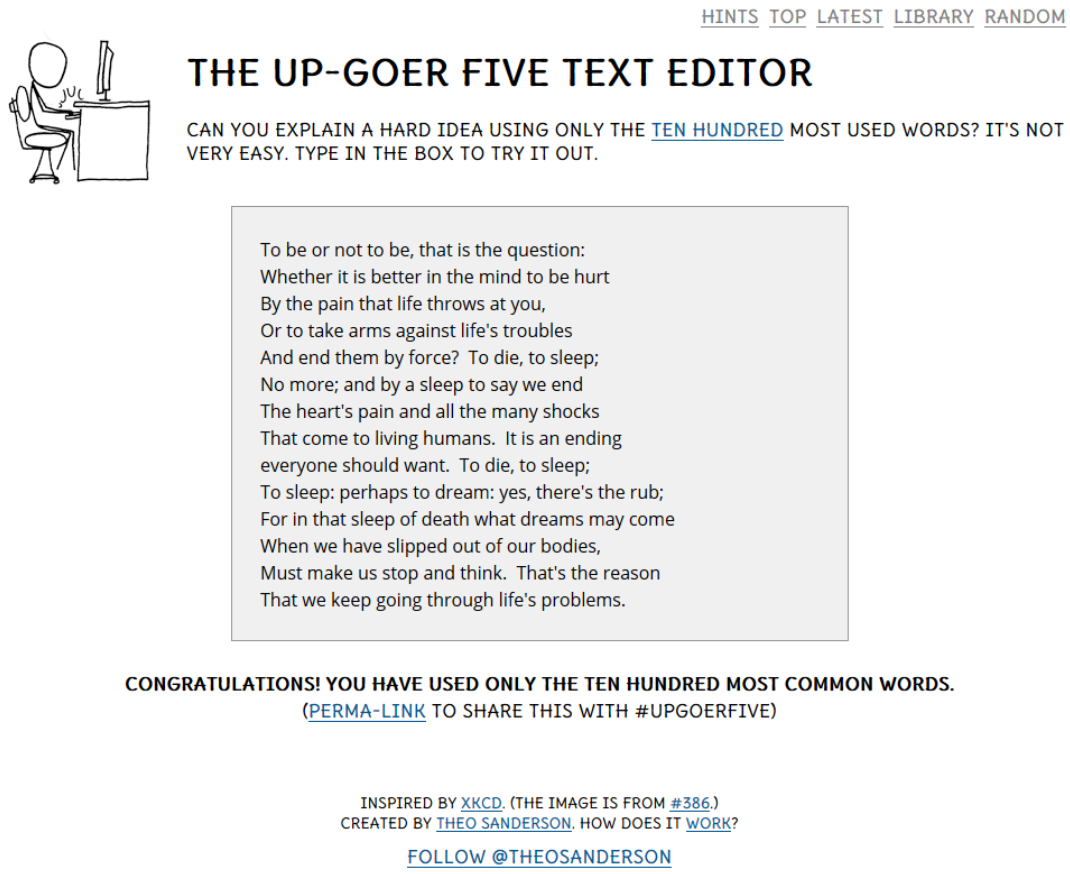
Pre spracovanie českého jazyka boli dostupné tri rôzne morfologické databázy pre Majku a to:

- dáta pre pridelenie základného tvaru (lemmy) a tagu slova analyzovanému slovnému tvaru
- dáta pre vygenerovanie všetkých tvarov sova a tagov zo zadaného základného tvaru slova (lemmy)
- dáta generujúce slovné tvary na základe zadanej lemmy alebo tagu

Tieto dáta pokrývajú 90% českého korpusu (okolo 500M slov). Obsahujú 3,393,080 trojíc zložených zo slovných tvarov + lemmat + tagov, čo predstavuje 903,888 odlišných slovných tvarov a 46,000 lemmat. Dáta tiež zahŕňajú časti takmer ôsmych biliónov zložených slov.

3.4 Návrh grafického rozhrania, funkcie a jednoduchosť

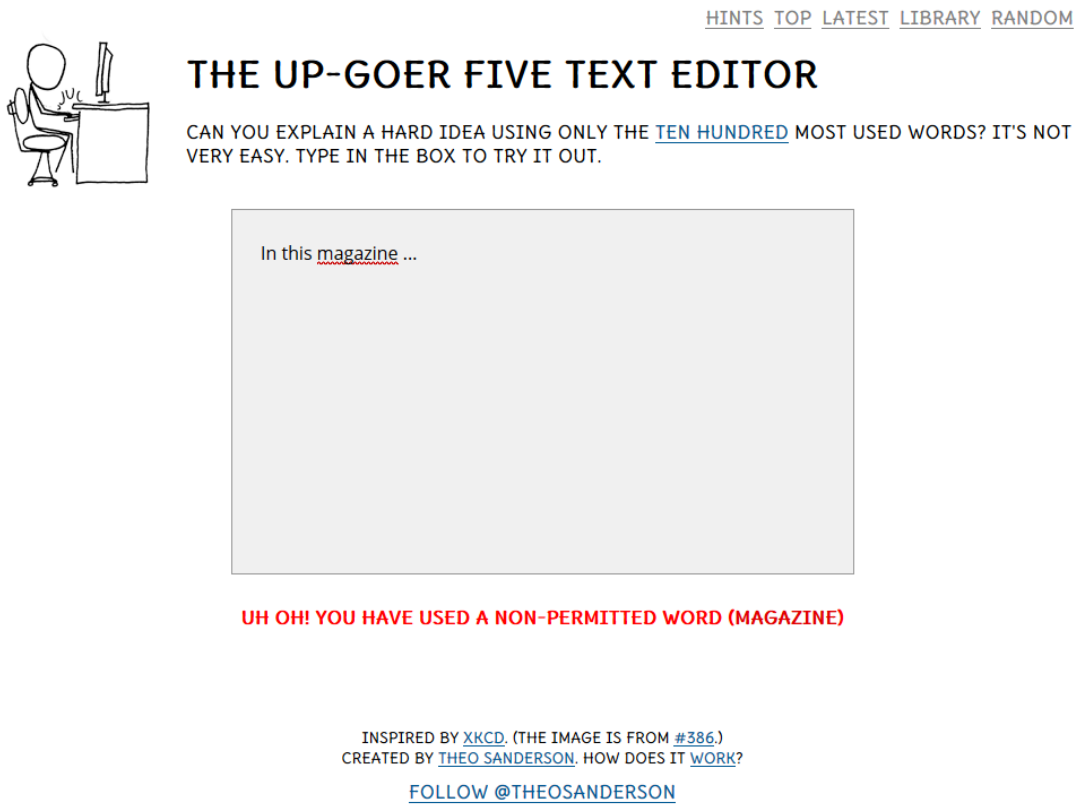
Pri návrhu tejto práce bolo mojím cieľom zachovať aplikáciu veľmi jednoduchú a priehľadnú. Ako pán Steve Krug spomína vo svojej knihe [6], „Nenechaj ma rozmýšľať – to je prvé pravidlo použiteľnosti pri vytváraní aplikácií.“ V najlepšom prípade by malo byť pri prvom



Obr. 3.2: Na obrázku vidieť, ako aplikácia skontrolovala užívateľov vstup a vyhodnotila že využil len tisíc najčastejšie používaných slov.

pohľade na webovú stránku zrejmé, ako sa dá použiť. Ak to však nie je úplne možné bolo by aspoň potrebné, aby bol v tejto stránke „nenápadne“ zahrnutý návod ako s ňou pracovať. „Nenápadne“ je myslené takým spôsobom, aby to užívateľovi obzrejmilo prácu so stránkou, no aby ho rôzne dlhé návody neodradili. Preto, ak je to nutné, je z môjho pohľadu vhodné vložiť niekoľko krátkych poznámok k použitiu stránky napríklad ako pomôcku, ktorá sa zobrazí po nájdení myšou na jednotlivé elementy stránky. Vzhľad zobrazených elementov na stánke (ako je veľkosť, farba a rozostavenie), ich správne vybrané názvy a malé množstvo pozorne upraveného textu, by spolu mali tvoriť správny základ k porozumeniu takmer bez akejkoľvek námahy. Preto som celý návrh vytvárala tak, aby aplikácia vo výsledku čo najviac zodpovedala tomuto popisu. Inšpirovala som sa exisujúcim riešením, ktoré je uvedené v kapitole 3.2, a ktoré predstavuje veľmi jednoduché riešenie tohto problému. Stránka neobsahuje žiadne zbytočné rušivé prvky, vystihuje to jej účel. Okrem toho mi bol inšpiráciou aj Google. Je navrhnutý veľmi jednoducho a aj napriek tomu je jednou z najčastejšie používaných stránok na svete.

Pre jednoduchosť a jednoznačnosť mojej aplikácie som chcela zahrnúť len niekoľko funkcií. Jednou z najdôležitejších úloh tejto aplikácie je, aby užívateľ zadával vstup do nejakého textového bloku, kde ho môže editovať. Tento text, alebo vstup, musí byť následne porovnávaný s referenčným slovníkom a slová, ktoré sú nad hranicu niekoľkých najčastejšie používaných slov budú určitým spôsobom odlišené, aby bolo užívateľovi jasné, že dané slovo



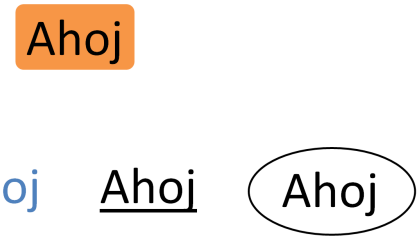
Obr. 3.3: Tento obrázok ukazuje funkciu vyhodnotenia užívateľského vstupu aplikácie, kde užívateľ nevyužil tisíc najčastejších, pretože použil slovo „magazine“, ktoré nepatrí do tejto hranice slov.

nespadá do vybraného rozsahu slov. Z tohto textového bloku bude tiež možné text kopírovať, respektíve ho tam vložiť. Aplikácia bude mať taktiež pomocné funkcie, ktoré budú umožňovať daný text uložiť ako textový súbor, respektíve načítať text z textového súboru.

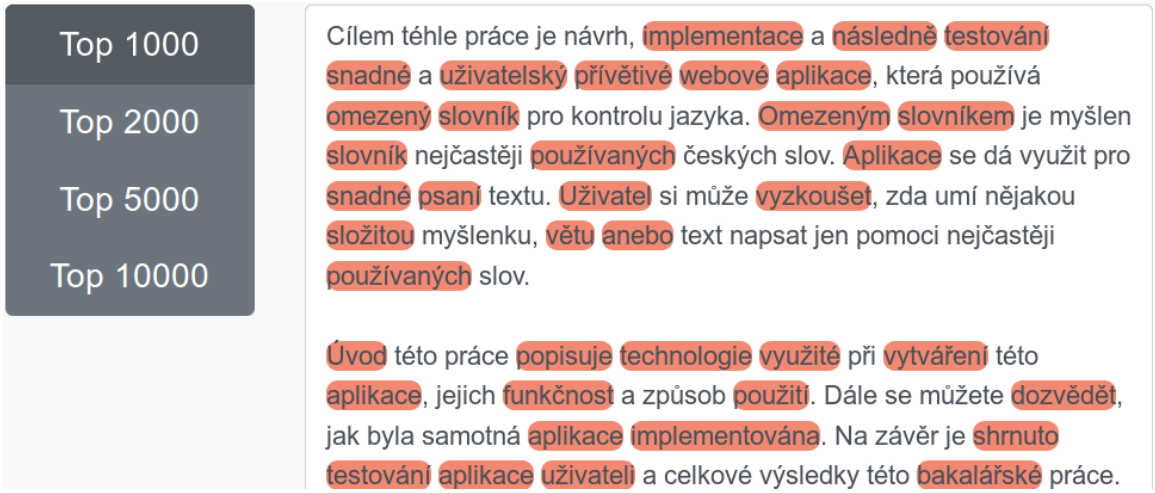
Na odlišenie slov, ktoré nie sú v rozmedzí niekoľkých najčastejších slov som si zvolila zvýraznenie pozadia za slovom. Učinila som tak po niekoľkých ukážkach pre možných užívateľov, kde to okrem tohto zvýrazňovania mohli porovnať aj s možnosťami ako podčiarknutie, zmenenie písma a zakrúžkovanie daného slova ako vidíte na obrázku 3.4. Všetci sa zhodli na zvýraznení, preto som túto variantu brala ako najviac prívetivú.

Pri zvýrazňovaní týchto slov som v prvotnom návrhu dané slovo zvýrazňovala automaticky pri písaní a teda ak niekto napísal napríklad prvé dva písmena zo slova, ktoré však nič neznamenali, boli tieto dve písmená rovno zvýraznené. Zvýrazňovanie tak prebiehalo dynamicky, no pri testovaní som zistila, že to na užívateľov nepôsobi veľmi príjemným dojmom. V českom jazyku totiž existujú rôzne predpony a preto pri tomto dynamickom zvýrazňovaní vznikal blikací efekt. To som následne upravila takým spôsobom, aby bolo dané slovo zvýraznené vtedy, keď sa v danom slove nenachádza kurzor, určujúci pozíciu v textovom okne. To daný problém vyriešilo.

Keďže každý užívateľ môže mať inú predstavu, ako veľmi obmedzný slovník chce použiť, tak som vytvorila riešenie tak, aby aplikácia poskytovala aspoň niekoľko možností, z ktorých si užívateľ môže vybrať. Navrhla som to ako výber z niekoľkých vopred stanovených možností. Pôvodne som uvažovala aj o posuvnej lište, kde si užívateľ vyberie ľubovoľnú hod-



Obr. 3.4: Na tomto obrázku vidieť rôzne druhy vyznačení, ktoré užívatelia porovnávali nie len na obrázku, ale aj v texte. Najviac sa im pozdávalo zvýraznenie pozadia slova.

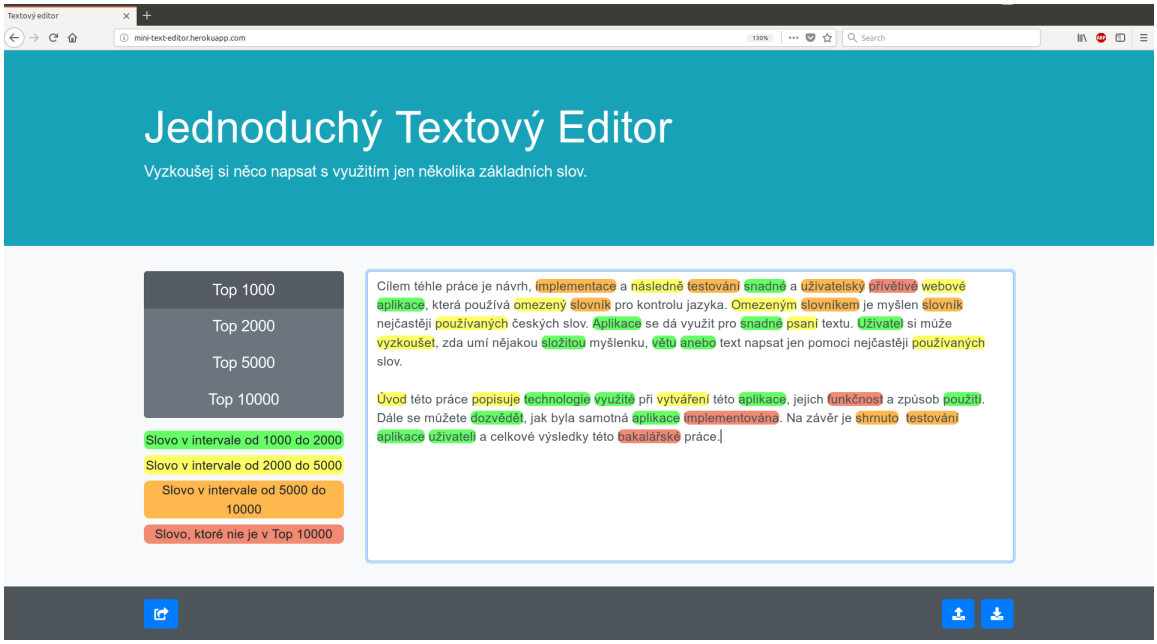


Obr. 3.5: Tetno obrázok slúži na porovnanie predchadzajúcej verzie aplikácie, ktorá používala len zvýraznenie jednou farbou.

notu počtu slov, no to by nemuselo spĺňovať cieľ aplikácie (slovník by už nebol obmedzený) a nie každý by vedel, ako s tým pracovať. Ak ma užívateľ vopred určené hodnoty slovníka, nemusí sa zamýšľať nad tým, aká hodnota by bola vhodná, stačí, že si vyskúša použitie nejakej z predvolených hodnôt.

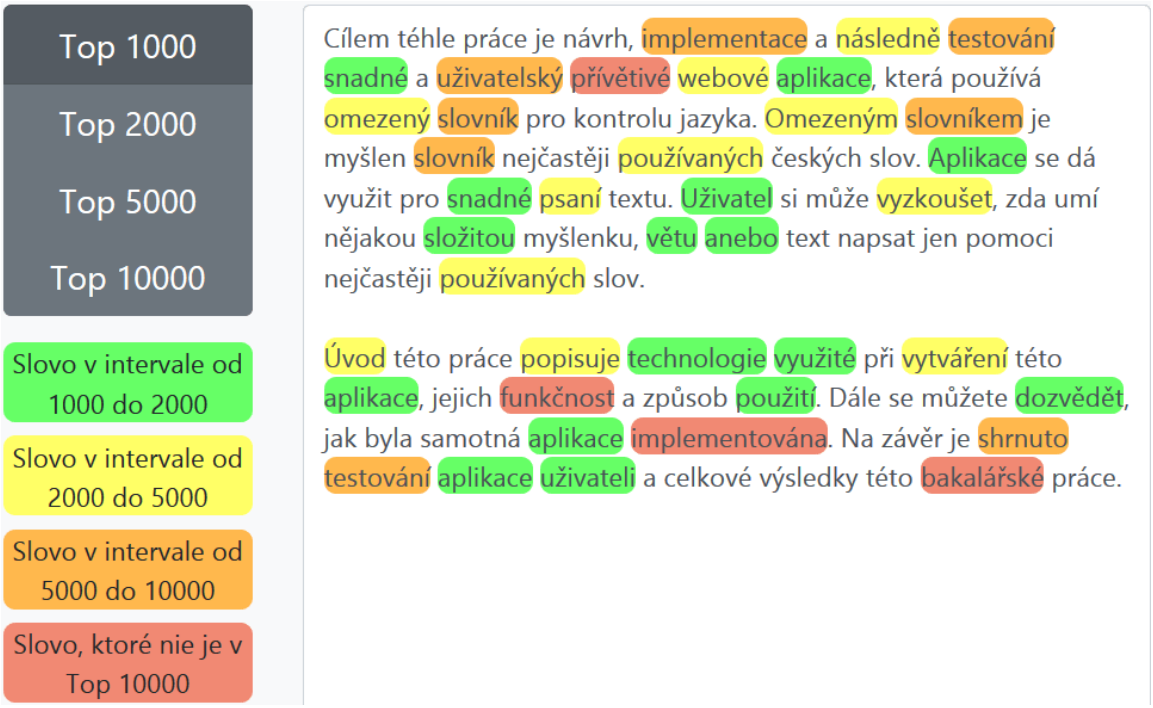
V prvotnom návrhu bolo zvýrazňovanie všetkých slov, ktoré nespĺňali zvolené obmedzenie označené rovnakým spôsobom. Po prvom testovaní som však od užívateľov dostala návrh na vylepšenie tejto možnosti tak, aby sa zvýraznenie menilo podľa toho, do ktorého intervalu dané slovo patrí. Tým pádom je užívateľovi pri písaní textu zrejmé, ako veľmi je dané slovo nepopulárne. Pôvodné riešenie môžete vidieť na obrázku 3.5.

Konečný návrh aplikácie je vidieť na obrázku 3.6 a 3.7. Okrem výsledného dizajnu stránky, môžete na daných obrázkoch vidieť aj funkčnosť aplikácie. Výsledná stránka teda obsahuje nádpis s popisom danej aplikácie a veľmi jednoduchý dizajn pre výber z možností ohraničenia slovníka. Súčasťou je textové pole pre užívateľský vstup. Okrem toho som vytvorila staticky zobrazené zápätie webovej stránky, v ktorom som netradičným spôsobom pridala možnosti na zdieľanie či inú prácu s aplikáciou. Učinila som tak preto, že väčšina užívateľov zdieľa stránky, či ich obsah až po tom, čo nadobudnú informáciu o tom, čo sa na danej stránke nachádza. Takto „oskenujú“ danú stránku, zistia ako funguje a následne vidia možnosť zdieľania. Navrhla som to tak, aby bolo zápätie zobrazené na fixnej pozícii, pretože v tejto jednoduchej aplikácii nie je jeho cieľom plniť obyčajnú úlohu päty dokumentu, ale zobrazuje ďalšie možnosti pre samotnú prácu s aplikáciou. Vďaka tomuto statickému



Obr. 3.6: Tento obrázok zobrazuje výsledné riešenie.

zobrazeniu teda užívateľ nemusí prejsť pomocnou posuvnej lišty celú stránku len preto, aby sa dostal k ďalšej funkcionalite, ale naopak aj na menších zariadeniach sú tieto možnosti vždy dostupné a viditeľné.



Obr. 3.7: Na tomto obrázku je ukážka funkcionality aplikácie a spôsobu zvýraznenia slov. Ukážka obsahuje môj preložený abstrakt tejto práce bez úprav pre možnosť hranice slovníka stanovenou na 1000 najčastejších slov.

Kapitola 4

Implementácia, testovanie a zhodnotenie výsledkov

V tejto kapitole sú opisané postupy pri implementácii, od vytvárania slovníka (spracovávanie dát) až po zhotovenie celej aplikácie (implementácia prostredníctvom webových technológií). Taktiež sú opísané problémy, ktoré nastali a spôsoby, akými boli riešené.

4.1 Vytvorenie slovníka

Vytváranie slovníka prebiehalo v niekoľkých krokoch. Prvým krokom bolo získanie dát, ktoré budú predstavovať určité frekvenčné zoznamy českých slov. Následne bolo potrebné tento zoznam slov spracovať do takého tvaru, aby sa s nim vo výsledku dalo jednoducho pracovať. Podrobnejší popis tohto kroku som uviedla v sekcii 3.3.

Pre vytvorenie koncového tvaru slovníka som si však potrebovala vytvoriť pomocný program, ktorý by zobral prvotný tvar slova (dostupný z frekvenčných zoznamov českých slov) a previedol ho na jeho základný tvar – lemma (v češtine sa jedná napríklad o nominatív jednotného čísla či slovesný neurčitok). Práve pri tomto prevode som v implementácii využila program Majka. Konkrétny popis jeho funkčnosti je spomínaný vyššie v podsekcii 3.3.1.

Zo získaných základných tvarov som však potrebovala ešte nadobudnúť všetky tvary daného slova, vzhľadom k tomu, že v češtine existujú rôzne tvary slov na základe ich skloňovania či časovania. Aj v tomto prípade mi pomohol program Majka, pomocou ktorého som tak zo vstupnej lemmy získala všetky tvary daného slova. Tieto získané dáta som rozdelila do výsledného slovníka tak, aby bola následná práca zjednodušená a to umiestnením jedného slova so všetkými jeho tvarmi na jeden riadok daného súboru. Ukážka časti slovníka je zobrazená na obrázku 4.1.

Tento pomocný program pre spracovanie slovníka som implementovala pomocou jazyku Java, ktorý má veľmi užitočne spracované použitie regulárnych výrazov pre prácu so znakovými reťazcami. Taktiež je možné využiť funkciu prostredníctvom `ProcessBuilder`-a, ktorá spúsi príkaz v terminály, čo bolo potrebné pre spluprácu s Majkou bez užívateľského zásahu a teda bez toho, aby som získané výsledky prevádzala ručne. Bližší opis `ProcessBuilder`-a je popísaný v podsekcii 2.4.3.

Konkrétny postup pri vytváraní tohto pomocného programu bol taký, že som si v prvom rade vytvorila triedu, v ktorej bola obsiahnutá hlavná funkcia, ktorá bola vykonaná po spústení programu. Obsah hlavnej funkcie v sebe zahrňuje nasledujúce úlohy:

```
73 nový nových novým novýma novými novou nová nové
   nového novém novému novější novějších novějšího
   novějším novějšíma novějšími novějšímu noví nenový
   nenových nenovým nenovýma nenovými nenovou nenová
   nenové nenového nenovém nenovému nenovější
   nenovějších nenovějšího nenovějším nenovějšíma
   nenovějšími nenovějšímu nenoví nejnovější
   nejnovějších nejnovějšího nejnovějším nejnovějšíma
   nejnovějšími nejnovějšímu nejnenovější nejnenovějších
   nejnenovějšího nejnenovějším nejnenovějšíma
   nejnenovějšími nejnenovějšímu
74 první prvních prvního prvním prvníma prvními prvnímu
75 její jejích jejího jejím jejíma jejími jejímu
76 jiný jiných jiným jinýma jinými jinou jiná jiné
   jiného jiném jinému jiní nejiný nejiných nejiným
   nejinýma nejinými nejinou nejiná nejiné nejiného
   nejiném nejinému nejiní
```

Obr. 4.1: Tento obrázok zobrazuje umiestnenie jednotlivých slov v spracovanom referenčnom slovníku pre aplikáciu. Každé slovo aj so všetkými jeho tvarmi je umiestnené v osobitnom riadku.

- vytváranie zoznamu príkazov, ktoré sa následne majú spustiť v operačnom systéme (navrhnuté pre Linuxový operačný systém Ubuntu),
- načítanie vstupného súboru po riadkoch,
- vytvorenie pomocného súboru s obsahom lemmat,
- vytvorenie finálneho súboru referenčného slovníka,
- spúšťanie príkazov z vytvoreného zoznamu príkazov.

Na vytváranie príkazov, ktoré budú spúšťané systémovo, som si vytvorila pomocnú metódu (`public static List<String> createCommand(String command)`), ktorej vstupný parameter bol String, v ktorom bol daný príkaz obsiahnutý. Z tohto vstupného parametra som však potrebovala získať iný typ – `ArrayList`, ktorý v sebe obsahoval predvolené hodnoty, a to: ktorý program sa má spúšťať (`bash`), s akými možnosťami sa má spustiť (`-c`) a následne samotný príkaz zo zadaného parametra. Takto vytvorený príkaz bol návratovou hodnotou tejto metódy.

Na spúšťanie príkazov som taktiež vytvorila pomocnú metódu (`public static void executeCommands(List<List<String>> commands)`). Vstupným parametrom je zoznam všetkých príkazov. V danej metóde som pre všetky obsiahnuté príkazy pomocou lambda výrazu vytvorila `ProcessBuilder`, ktorý dané príkazy spúšťal. Ukážku tejto pomocnej metódy môžete vidieť na obrázku 4.2.

Beh programu bol následovný: v prvom rade som si vytvorila zoznam príkazov, ktorý bol typu `ArrayList<List<String>>`. Do tohto zoznamu som vkladala príkazy, ktoré sa následne majú spustiť. Vytvorila som si tak pomocné súbory, prostredníctvom príkazu `touch temporary.txt`, kde `temporary.txt` je názov vytváraného súboru. Tieto príkazy som spúšťala v priebehu programu a preto som potrebovala využiť aj Java metódu na vyprázdnenie daného listu príkazov (`.clear()`). Následne som načítala vstupný súbor – frekvenčný zoznam českých slov do riadkov a pomocou regulárnych výrazov v Jave som preformátovala jednotlivé riadky tak, aby som z nich získala len jednotlivé slová. Tieto slová však ešte

```java
// method to execute commands in list passed as argument
public static void executeCommands(List<List<String>> commands) {
    commands.forEach(command->{
        try {
            Process runCommand = new ProcessBuilder(command).start();
        } catch (IOException e) {
        }
    });
}
```

Obr. 4.2: Na obrázku sa nachádza vytvorená metóda, ktorej úlohou je postupne spustiť všetky príkazy obsiahnuté vo vstupnom parametri. Využíva pri tom lambda výraz a `ProcessBuilder`.

neboli všetky v ich základných tvaroch. Preto som pre každé z nich musela vytvárať príkaz, ktorý spúšťal pomocný program Majka a získal tak základné tvary slova, ktoré som uložila do pomocného súboru. Následne som z vytvoreného pomocného súboru načítavala jednotlivé základné tvary a taktiež vytvorila príkazy, ktoré spúšťali Majku na získanie všetkých tvarov pre danú lemmu a výstup ukladali do druhého pomocného súboru. Vzhľadom k tomu, že vstupné frekvenčné zoznamy boli veľmi rozsiahle a obsahovali slová, ktoré samotný program Majka nepoznal, mohol tento výstupný súbor obsahovať niektoré riadky prázdne. Z daného súboru som preto prázdne riadky odstránila a vytvorila tak konečný tvar referenčného slovníka.

4.2 Konfigurácia frameworku Symfony

Aplikáciu som sa rozhodla naprogramovať pomocou frameworku Symfony, pretože sa stáva čím ďalej tým viac populárnym. Začínala som ako začiatočnik, no jeho používanie nie je náročné, čo bolo veľké pozitívum. Pracovala som v operačnom systéme Ubuntu (distribúcia operačného systému Linux), v ktorom bolo jednoduché nakonfigurovať Symfony prostredníctvom programu *Composer*. Tento program spravuje závislosti v jazyku PHP. Umožňuje deklarovať knižnice, na ktorých závisí projekt budovaný v jazyku PHP.

Za pomoci programu *Composer* som teda vytvorila Symfony projekt, ktorý bol vygenerovaný priamo s kostrou programu a predprípravenou demo aplikáciou. Tieto dve skutočnosti uľahčovali prácu s frameworkom Symfony, vzhľadom k tomu, že ukazovali odporúčaný spôsob pre vývoj aplikácií. Okrem toho taktiež obsahovali okomentovaný kód, čo mi ako začiatočníkovi pomohlo sa rýchlo zorientovať vo funkčnosti tohto frameworku.

Vďaka tomu, že bola automaticky vygenerovaná celá štruktúra demo aplikácie, dalo sa hneď sústrediť na vývoj vlastnej aplikácie len jednoduchým nahradením obsahu niektorých súborov za vlastný.

4.3 Implementácia webovej stránky – Controller, Twig

Vzhľadom k tomu, že som danú aplikáciu implementovala vo frameworku Symfony, samotná webová stránka bola vytvorená pomocou šablóny v šablónovacom systéme *Twig*. Ako som už spomínala vyššie (kap. 2.3.1), hlavnými jazykovými prvkami tohto systému sú: prvok `{{ }}`, ktorý zobrazí výsledok vyhodnoteného výrazu a potom prvok `{% %}`, ktorý vykoná

daný príkaz. Súbory *Twig* boli po vytvorení projektu umiestnené v adresári `templates` (šablóny).

Pred tým, ako som začala vytvárať stránku pomocou *Twig*, bolo potrebné vytvoriť *Controller* (ďalej kontrolér), ktorý kontroluje chod samotnej aplikácie. Kontrolér môže byť akýkoľvek typ PHP operácie, ktorá môže byť volaná z kódu, ako napríklad funkcia či metóda. Väčšinou však predstavuje metódu v triede, ktorá daný kontrolér vytvára. Pracuje tak, že číta informácie z objektu `Request` (požiadavka) a vytvára návratový objekt `Response` (odpoveď). Týmto návratovým objektom môže byť HTML stránka, stiahnutie súboru, presmerovanie, chyba `404`, alebo čokoľvek iné. Kontrolér spúšťa akúkoľvek ľubovoľnú logiku, ktorú vyvíjaná aplikácia potrebuje pre poskytnutie obsahu stránky[1].

V svojej aplikácii som práve prostredníctvom kontroléra vytvorila návratový objekt, ktorý presmeroval stránku na súbor *Twig*. Presmerovanie viedlo na súbor, v ktorom som neuvádzala implementáciu HTML kódu, ale v ktorej som len pomocou prvkov tohto šablónovacieho systému vytvorila úseky kódu, ktoré nahradzovali niektoré časti vytvorenej HTML stránky. Aby to bolo možné, samotná HTML stránka niekde uvedená byť musela. Vytvorila som teda dva súbory *Twig*. Jeden predstavoval odkazovaný súbor z Controllera, ktorý obsahoval len prvky systému *Twig* a druhý súbor, v ktorom bola implementovaná stránka HTML a v ktorej som dané prvky z odkazovaného súboru využívala. Keďže však odkazovaný súbor neobsahoval požadovaný obsah na zobrazenie, bolo potrebné na začiatok tohto súboru uviesť, že rozširuje obsah iného dokumentu. Konkrétne príklady použitia systému *Twig*:

- `{% extends 'base.html.twig' %}` – rozšírenie základného súboru, ktorý obsahuje HTML kód. Toto rozšírenie je uvedené v odkazovanom súbore z kontroléra.
- `{% block body %} ... {% endblock %}` – predstavuje blok HTML kódu, ktorý sa následne bude dať použiť v rozšírovanom súbore skrátenou formou a to `{% block body %}{% endblock %}` bez uvádzania obsahu. Tento obsah je automaticky nahradený predošlou špecifikáciou.
- `{% block javascripts %} ... {% endblock %}` – tento blok obsahuje odkazy na používané skripty pre prácu s rôznymi knižnicami ako je jQuery či Bootstrap.

Pomocou definície týchto blokov sa dá hlavný rozšírovaný kód udržiavať priehľadný. Okrem toho sa to dá využiť aj vtedy, ak by vývojár vytváral stránku s konštantnou stavbou hlavičky a päty stránky, ktorá by bola prostredníctvom kontrolérov z rôznych odkazov doplnená o rôzne telo dokumentu. Definovať tieto bloky je možné aj priamo v hlavnom rozšírovanom dokumente a po ich definícií môžu byť používané v skrátenej forme `{% block NÁZOV_BLOKU %}{% endblock %}` už bez ich obsahu, ktorý sa v definícii uvádza medzi týmito dvoma prvkami predstavujúcimi začiatok a koniec daného bloku.

Takýmto spôsobom som vytvárala túto aplikáciu. Príklady výslednej stránky sú uvedené v kapitole o návrhu 3.4.

4.4 Štýl stránky – CSS a Bootstrap

Grafickú časť stránky som vytvárala prostredníctvom dvoch nástrojov a to jazyka CSS a frameworku Bootstrap. Na štýl celej aplikácie som používala predovšetkým Bootstrap, CSS som využila len pre zosynchronizovanie dvoch elementov, pre zvýrazňovanie slov pri písaní

[1] Controller (Symfony Docs) – https://symfony.com/doc/current/controller.html

do textového okna. Jedná sa o element `<div>` a element `<textarea>`, vzhľadom k tomu, že priamo v elemente `<textarea>` nie je možné zvýrazňovať jednotlivé slová a preto sa pod ňou nachádza element `<div>` s rovnakým chovaním, ktorý je možné rozšíriť o `<span>` elementy. Týmto elementom som následne pridávala triedu, ktorej som v CSS dokumente pridala pravidlo pre štýl zvýraznenia a to zmenu farby pozadia, čo spôsobuje samotné zvýraznenie. Ak by užívatelia mali pripomienky k spôsobu zvýrazňovania, túto možnosť by bolo v danom CSS pravidle jednoduché modifikovať. Bolo by možné zmeniť farbu zvýraznenia či dané slovo podčiarknuť a tak ďalej.

Prostredníctvom knižnice Bootstrap som definovala celkové umiestnenie elementov na stránke, responzívne správanie sa aplikácie na rozličných zariadeniach, skupinové tlačidlá, špeciálne záhlavie a využila som mnoho užitočných preddefinovaných prostriedkov, ktoré uľahčili vytvorenie dizajnu danej aplikácie.

4.5 Hlavná funkcionalita – JavaScript

Hlavné funkcie, ktoré aplikácia obsahuje, som implementovala v jazyku JavaScript. Medzi tieto funkcie patrí zvýrazňovanie slov v textovej oblasti na základe porovnania s referenčným slovníkom, stiahnutie a načítanie súboru z počítača, zdieľanie odkazu na stránku prostredníctvom Facebooku alebo mailu, zmeny zvýraznenia slov po odkliknutí inej varianty slovníka a ďalšie.

Hlavnou funkciou je odlišenie slov, ktoré nezodpovedajú slovám z obmedzeného slovníka. Aby bolo možné túto funkciu implementovať, potrebovala som zosynchronizovať správanie dvoch HTML elementov a to blokového elementu `<div>` a elementu `<textarea>`. Textarea slúži na samotné písanie, kopírovanie a vkladanie textu, zatiaľ čo v elemente div používam ďalší pomocný element `<span>`, ktorému prideľujem farbu pozadia, čo spôsobí samotné zvýraznenie. Bolo tiež nutné zaistiť, aby sa tieto dva elementy nie len chovali rovnako, ale mali rovnakú pozíciu na stránke. To som dosiahla pomocou jazyka CSS a prostredníctvom spracovávateľa udalostí (event handler).

Samotná funkcia bola v skratke implementovaná tak, že som v prvom rade kontrolovala vstup užívateľa a testovala, či sú jednotlivé slová oddelené bielymi znakmi obsiahnuté v obmedzenom slovníku a ak sa v danom slovníku slovo nevyskytovalo, ukladala som ho do zoznamu slov na zvýraznenie. Tento zoznam som následne predávala ďalšej funkcii, ktorá prerábala obsah elementu `<div>` tak, že vyhľadala slová, ktoré sa majú nahradiť v danom texte a nahradila všetky výskyty daných slov za text, ktorý obsahoval element `<span>` so špecifikovanou triedou pre previdlá CSS. Napríklad slovo ahoj, ktoré sa v slovníku vyskytovalo až na hranici do 10000 slov, bolo pri nižších hraniciach slovníka nahradené takto: `<span class='highlight-span'>ahoj</span>`. Tento element v sebe obsahoval dané slovo, ktoré bolo nahradzované. Následne som prepísala obsah elementu `<div>` prostredníctvom funkcie jazyka JavaScript a jeho knižnice jQuery s využitím selektora a príkazu *.html()* následovným spôsobom: (`$("SELEKTOR").html("TEXTOVÝ REŤAZEC, ktorý môže obsahovať HTML prvky");`).

Pre zosynchronizovanie týchto dvoch elementov – `<div>` a `<textarea>`, bolo tiež potrebné posúvať ich obsah vertikálnym smerom pomocou posuvnej lišty súčasne. Pôvodne som chcela zachovať posuvnú lištu aj pre horizontálny smer, no nastal problém pri odsadzovaní vnútorného textu týchto dvoch elementov, kde sa na konci riadku slová ocitali na rôznych pozíciách ako napríklad umiestnenie koncového slova v textovom bloku elementu `<textarea>` bolo v elemente `<div>` na začiatku nasledujúceho riadku. Ako najlepšie riešenie mi pripadalo možnosť horizontálneho skrolovania odstrániť, vzhľadom k tomu, že v češtine

neexistuje natoľko dlhé slovo, aby to bolo vôbec nutné posúvať pomocou horizontálnej posuvnej lišty. Ak by však niekto takéto slovo predsa len vymyslel, prostredníctvom CSS sa na základe definovaného pravidla v obidvoch elementoch toto slovo na konci riadku odsekne do riadku nasledujúceho.

Ďalej bolo dôležité, aby sa zvýraznenie menilo na základe zvoleného slovníka. Preto som pridala funkcionalitu na spracovanie udalostí (*event listener*) pre všetky možnosti slovníka a na základe výberu sa daný slovník nastaví. Okrem toho som v novej verzii implementovala zmenu nastavenia zvýraznenia, kde som pre jednotlivé intervaly slovníka, z ktorého zvýrazňované slovo pochádza, zmenila farbu pozadia daného slova. Tieto farby som vyberala tak, aby jasne určovali, do ktorej skupiny slovo patrí a teda napríklad slová, ktoré sú v rozmedzí od 1000 do 2000 najčastejších slov, budú označené zelenou farbou (patrí ešte k veľmi často používaným slovám) a slova nad 10000 sa označia červenou farbou (nespĺňa ani najmenej obmedzený výber).

Aplikácii som okrem týchto hlavných funkcií doplnila niekoľko menších, ako načítavanie súboru, respektíve uloženie textu ako súboru či zdieľanie danej stránky. Na ich implementáciu som taktiež využila jazyk JavaScript. Riešenia týchto funkcií nebolo náročné implementovať, mnoho z nich bolo možné vyhľadať na webových stránkach *Stack Overflow*[2], ktoré mi boli inšpiráciou pri ich implementácii.

Okrem príkladov aplikácie uvedených v kapitole 3.4 sa ďalšie ukážky použitia nachádzajú v prílohe A, ktoré som získala na základe užívateľských testov aplikácie.

4.6 Testovanie

Testovanie aplikácie sa vykonáva pre overenie a následné vylepšovanie akuálneho riešenia na zákade spänej väzby od užívateľov. Len pomocou testovania je možné zistiť, či daná aplikácia bude na webe užitočná a úspešná. V tom, kedy danú webovú stránku či aplikáciu vývojár testuje, je však taktiež rozdiel. Na testovanie väčšinou nepostačuje fáza, v ktorej má programátor vyvinutý len jednoduchý prototyp aplikácie [10]. Keď sa jedná o aplikáciu, je pre vykonávanie testov veľmi dôležité zaviesť do danej aplikácie minimálne požadovanú funkčnosť ešte pred samotným testovaním.

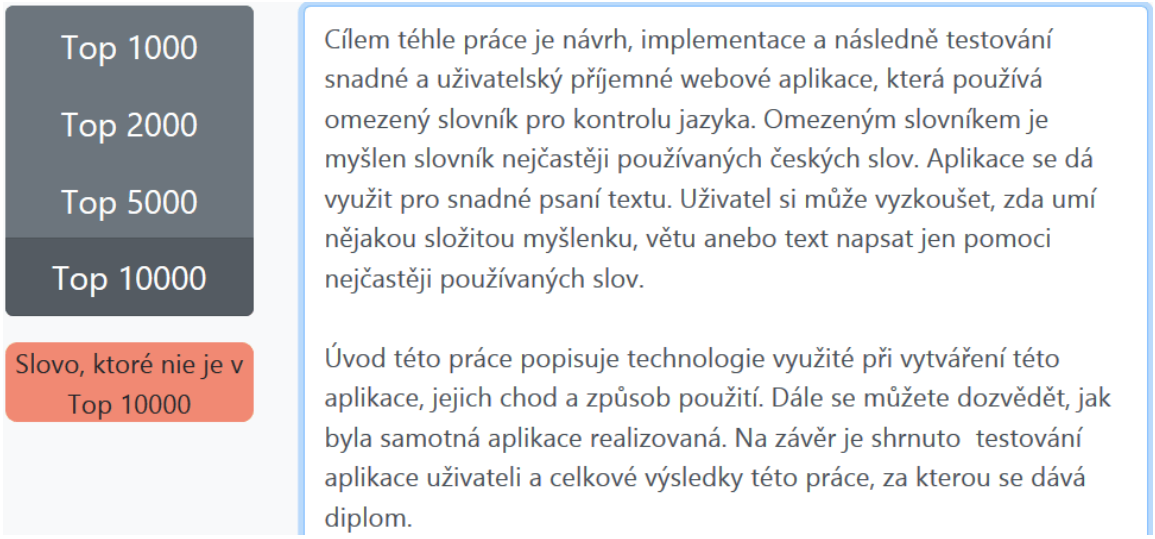
Svoju aplikáciu som si skúsila otestovať aj sama. Výsledok je zobrazený na obrázkoch 4.3 až 4.6. Je na nich zobrazená ukážka funkcionality, kde som sa snažila svoj abstrakt prepísať postupne pomocou jednotlivých možností. Preložený abstrakt bez úprav môžete vidieť v kapitole 3.4 na obrázku 3.7.

4.6.1 Testovanie riešenia s užívateľmi

Svoju aplikáciu som testovala dvomi spôsobmi. Jeden zahŕňal interaktívne testovanie aplikácie s užívateľmi, druhý spôsob bol nepriamou komunikáciou s užívateľmi, kde som im poslala dokument na vyplnenie spolu s dotazníkom.

Pri interaktívnom testovaní som si na užívateľoch všímala predovšetkým ich prácu s danou aplikáciou. Vzhľadom k tomu, že som sa snažila o jednoduchosť danej aplikácie, väčšina užívateľov reagovala veľmi rýchlo pri práci na danej stránke. Zorientovanie sa im trvalo len malú chvíľu. Pri prvom pohľade vedeli, že majú využiť textové okno pre ich užívateľský vstup. Vo svojej aplikácii som taktiež vytvorila prostredníctvom Bootstrapu možnosť zobrazovania nápovedy, čo užívateľom uľahčilo prácu s danou stránkou. Na dizajn reagovali

[2]Stack Overflow - Where Developers Learn, Share, & Build Careers – https://stackoverflow.com/

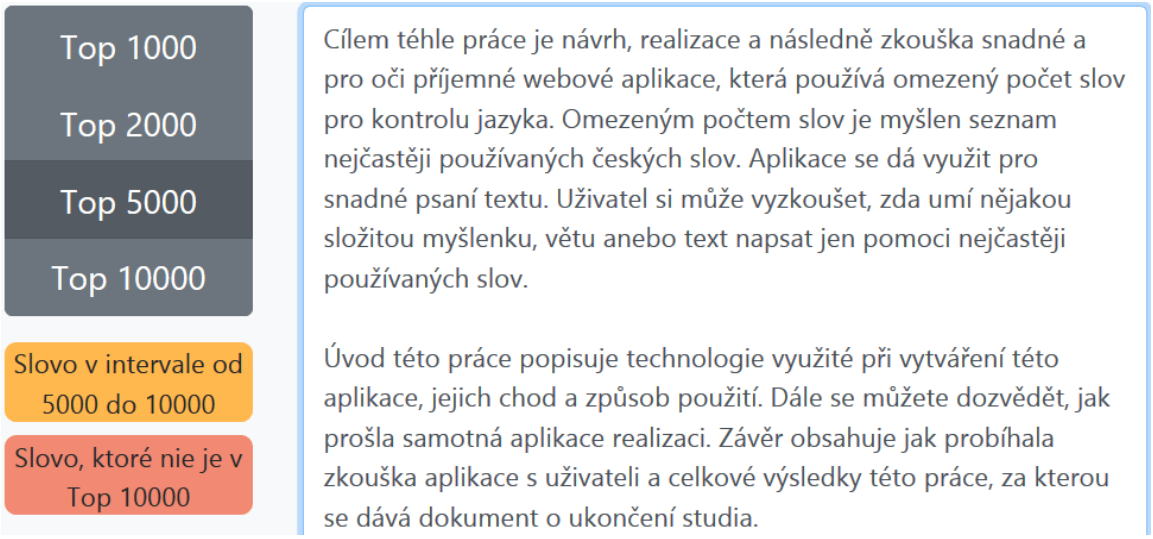


Obr. 4.3: Na obrázku je zobrazená funkcionalita daného riešenia s obsiahnutým abstraktom tejto práce a úpravou na 10000 najčastejších slov.

tiež pozitívne. Nie všetci si však všimli možností na konci stránky, ktoré obsahovali zdieľanie aplikácie, nahranie súboru z lokálneho úložiská či stiahnutie písaného textu v danom textovom okne. Mohlo to byť spôsobené práve nie veľmi tradičným návrhom stránky, ktorý som podrobnejšie opísala v kapitole 3.4. Bola to však len menšina užívateľov, preto by som to pri prehodnotení návrhu nepovažovala za nevhodnú možnosť, ale práve naopak. Napriek tomu by som zvážila otestovať umiestnenie týchto možností na vrchnú časť stránky, čo je v dnešnej dobe zaužívaný štandard. Ak by testy dopadli rovnako (menšina užívateľov si dané možnosti nevšíma), považovala by som obe možnosti umiestnenia týchto prvkov za rovnako správne. To, že si niektorí užívatelia tieto možnosti nevšimli, mohlo byť taktiež ovplyvnené ich záujmom a ochotou.

Druhá časť testovania prebiehala testovaním aplikácie nepriamym kontaktom s užívateľmi, ktorým som poslala dokument s dotazníkom na vyplnenie. V tomto dokumente sa nachádzal odkaz na stránku a popis postupu práce, ktorej výsledky som chcela získať. Jednalo sa o otestovanie aplikácie tým, že ju budú používať. Dala som im za úlohu nájsť, alebo napísať určitý odstavec textu, kde by si v prvom rade nevšímali dané zvýraznenia. Potom som ich požiadala o postupné prepísanie daného odstavca prostredníctvom možností ohraničenia tak, aby daný text obsahoval čo najmenej zvýraznených častí. Výsledky tohto testovania sa nachádzajú v prílohe A. Okrem toho som priložila dotazník, ktorým som sa chcela dozvedieť predovšekým odpovede na nasledujúce otázky:

- Ako sa užívateľovi pracovalo s danou aplikáciou, či mu bolo všetko zrejmé.
- Aká bola hranica slovníka, kedy začal mať užívateľ problémy s prevodom.
- Čím mohla byť táto hranica ovplyvnená (rodný jazyk slovenčina, spôsob prevodu bez využitia pomôcok a iné).
- Akú majú predstavu o využití danej aplikácie.
- Ako by zhodnotili dizajn aplikácie a ich celkové hodnotenie aplikácie.



Obr. 4.4: Na obrázku je zobrazená funkcionalita daného riešenia s obsiahnutým abstraktom tejto práce a úpravou na 5000 najčastejších slov.

- A koľko času im testovanie zabralo (aby som vedela, ako veľmi sa im potrebujem odvďačiť).

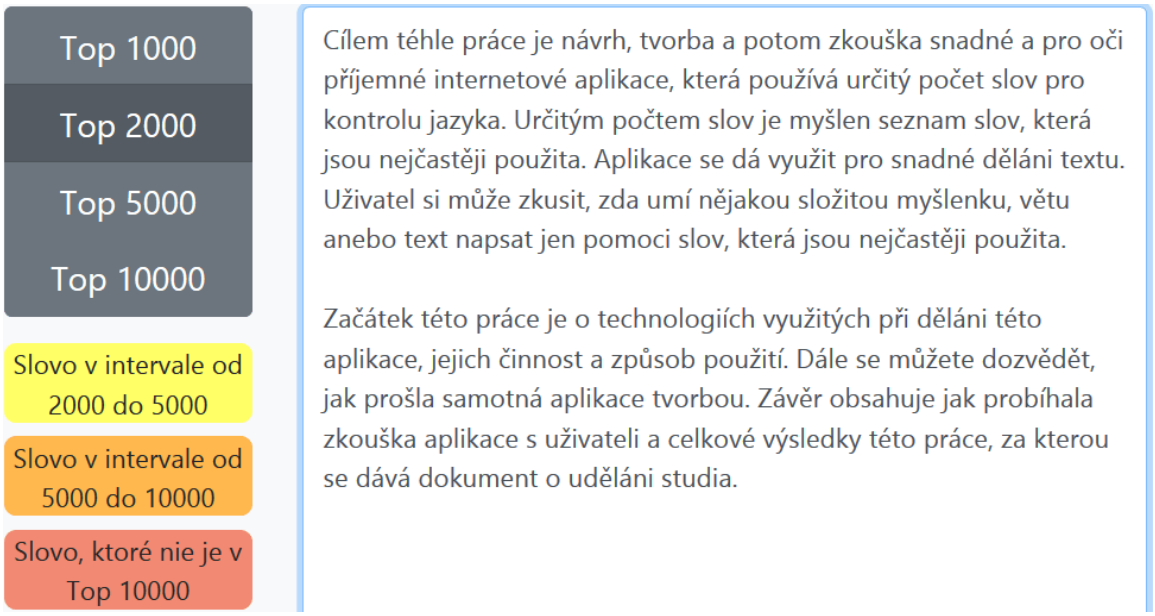
Výsledné reakcie užívateľov boli vcelku pozitívne, no mnoho z nich nemalo predstavu o využití danej aplikácie. Dizajn aplikácie na nich pôsobil príjemným dojmom, chválili nenáročnosť aplikácie. Medzi najčastejšiu hranicu obmädzenia slovníka, s ktorou sa im prepisovanie textu zdalo už náročnejšie bola pre Slovákov hranica 5000 slov, pre Čechov to bolo 2000 slov. Mnohí poznamenali, že vlastné názvy osôb či vecí sa nesnažili po zvýraznení prepisovať, taktiež navrhli, že by mohla aplikácia tieto názvy rozlíšiť a nezvýrazniť ich. Mnoho užívateľov na prevod slov používalo synonimický slovník českého jazyka. Priemerná doba týchto testov sa pohybovala okolo 30 minút.

4.6.2 Testovanie v rôznych prehliadačoch a na rôznych zariadeniach

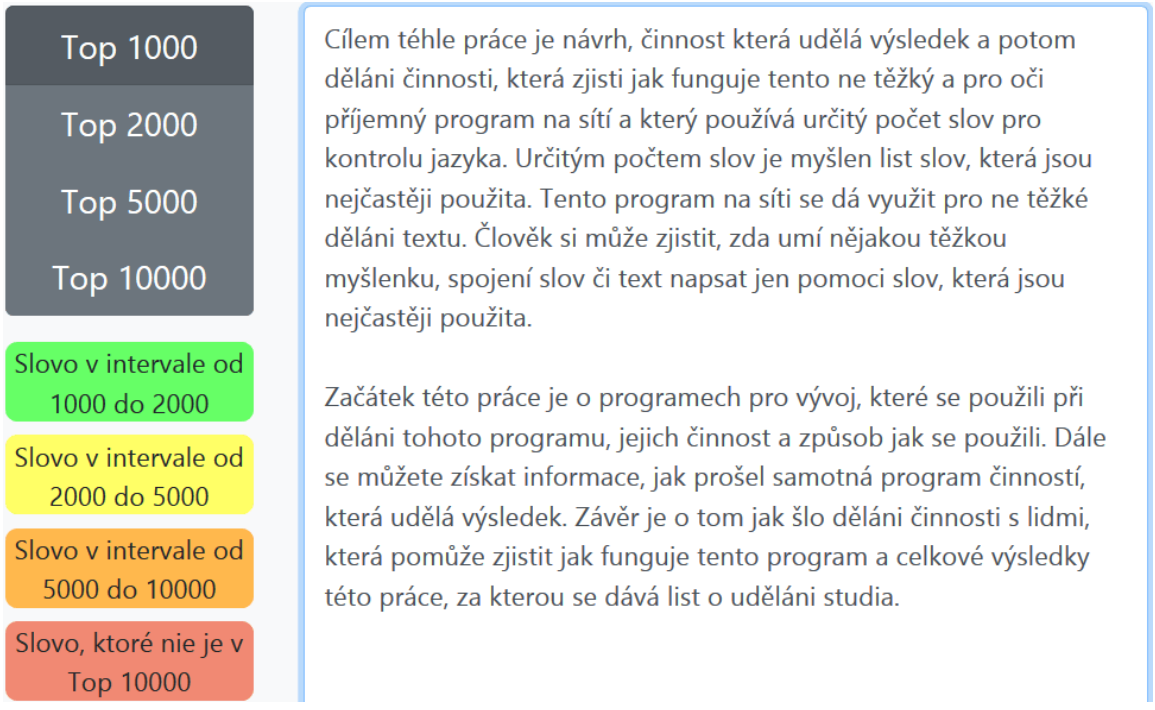
Okrem užívateľských testov som danú aplikáciu testovala aj v jej responzívnosti, či už na rôznych zariadeniach, alebo prostredníctvom rôznych prehliadačov.

Kontrolovala som funkcionalitu na počítačových aj mobilných zariadeniach. Aplikáciu som vyvíjala tak, aby sa poziciovanie elementov na stránke menilo aj na základe šírky okna daného zariadenia. Preto sa mi podarilo vytvoriť prispôsobivé riešenie fungujúce na rôznych zariadeniach. Ako vyzerá webová aplikácia na počítačových zariadeniach je zobrazené v kapitole 3.4. Pre porovnanie je na obrázku 4.7 zobrazené, ako aplikácia vyzerá na mobilnom telefóne.

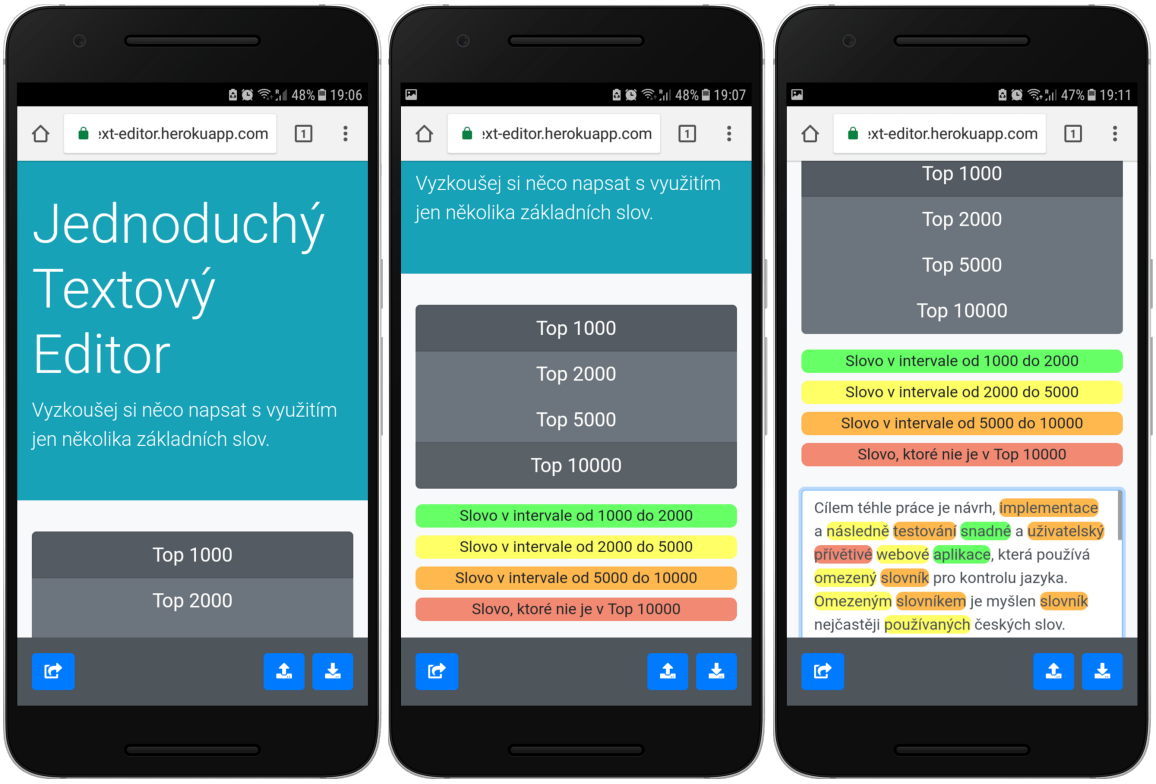
Aplikáciu som testovala v piatich webových prehliadačoch a overila som tak jej funčnosť v každnom z nich. Boli to následujúce webové prehliadače: Mozilla Firefox, Opera, Google Chrome, Microsoft Edge a taktiež ešte nie veľmi známy Vivaldi.



Obr. 4.5: Na obrázku je zobrazená funkcionalita daného riešenia s obsiahnutým abstraktom tejto práce a úpravou na 2000 najčastejších slov.



Obr. 4.6: Na obrázku je zobrazená funkcionalita daného riešenia s obsiahnutým abstraktom tejto práce a úpravou na 1000 najčastejších slov.



Obr. 4.7: Na tomto obrázku je zobrazené ako vyzerá dané riešenie na mobilnom zariadení.

Kapitola 5

Záver

Cieľom tejto práce bolo navrhnúť a vytvoriť jednoduchú webovú aplikáciu, ktorá bude pracovať s obmedzeným slovníkom a povoľovať editáciu textu na základe tohto slovníka. Hlavnými úohami boli: návrh danej aplikácie, jej implementácia a následne testovanie, čo patrí taktiež medzi hlavné úlohy pri vývoji akéhokoľvek užívateľského rozhrania. Mojou snahou bolo predovšetkým vyvinúť aplikáciu veľmi jednoduchú na používanie, čo sa mi podľa ohlasov testujúcich podarilo.

Webová stránka neobsahuje zbytočne veľké množstvo elementov, aby nepôsobila náročným dojmom. Pri zavádzaní funkcionality do aplikácie som chcela zahrnúť len primerané množstvo akcií, ktoré môže užívateľ na stránke previesť, pretože som nechcela mnohými funkciami zatieniť hlavný zmysel tejto aplikácie, ktorý spočíva v jej jednoduchosti a zameraní sa na používanie českého jazyka s obmedzením na niekoľko najčastejšie používaných slov.

V ďalšom vývoji tejto aplikácie by bolo možné doplniť niekoľko funkcií tak, aby si daná aplikácia zachovala aj svoju jednoduchosť, no taktiež by spĺňala niekoľko požiadavok užívateľov, ktoré som získala pri jej testovaní a to napríklad:

- Pridať k obmedzenému slovníku taktiež pomocný slovník s českými vlastnými menami, na základe ktorého by dané slová neboli zvýraznené.
- Možnosť výberu niekoľkých odborových slovníkov, ktoré by k slovníku s najčastejšie používanými slovami pridali do popredia najčastejšie používané slová z rôznych odborov (napríklad informatika, chémia či zoológia).
- Automaticky návrh synoným k slovám, ktoré sú zvýraznené.

Na základe týchto modifikácií by aplikácia v budúcnosti mohla nadobudnúť väčšiu použiteľnosť, no napriek nim by si taktiež zachovala svoju jednoduchosť.

Pri riešení tejto práce som načerpala mnoho vedomostí ohľadom vývoja webových aplikácií a užívateľských rozhraní. Obohatila som sa o skúsenosti v používaní frameworku Symfony a knižnice Bootstrap. Taktiež som sa zdokonalila v používaní iných moderných technológií ako sú JavaScript či jQuery.

Literatúra

[1] Chaffer, J.; Swedberg, K.: *Mistrovství v jQuery*. 2013, ISBN 9788025141038.

[2] Duckett, J.: *Beginning Web Programming with HTML, XHTML, and CSS*. Programmer to programmer, Wiley, 2004, ISBN 9780764570780.

[3] Flanagan, D.: *Java in a Nutshell*. Desktop quick reference, O'Reilly Media, Incorporated, 2005, ISBN 9780596007737.

[4] Gosling, J.; Joy, B.; Steele, G.; aj.: *The Java Language Specification*. Addison-Wesley Java series, Addison-Wesley, 2000, ISBN 9780201310085.

[5] Horstmann, C.: *Java SE8 for the Really Impatient: A Short Course on the Basics*. Java Series, Pearson Education, 2014, ISBN 9780133430196.

[6] Krug, S.: *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability*. Web Design Courses, Pearson Education, 2013, ISBN 9780133597264.

[7] Salehi, S.: *Mastering Symfony*. Packt Publishing, 2016, ISBN 9781784394264.

[8] Sharan, K.: *Beginning Java 8 Language Features: Lambda Expressions, Inner Classes, Threads, I/O, Collections, and Streams*. Expert's voice in Java, Apress, 2014, ISBN 9781430266594.

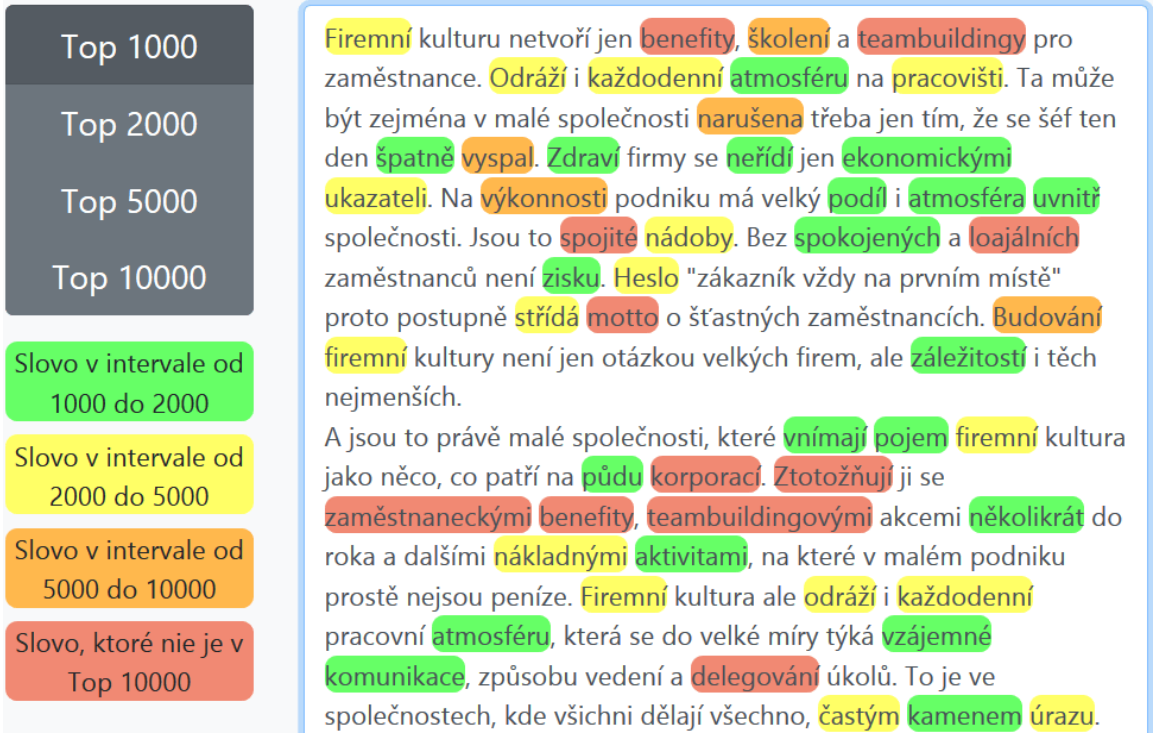
[9] Zaninotto, F.; Potencier, F.: *The Definitive Guide to symfony*. Apress, 2007, ISBN 9781430203797.

[10] Řezáč, J.: *Web ostrý jako břitva: Návrh fungujícího webu pro webdesignery a zadavatele projektů*. BAROQUE PARTNERS s.r.o., 2014, ISBN 9788087923016.

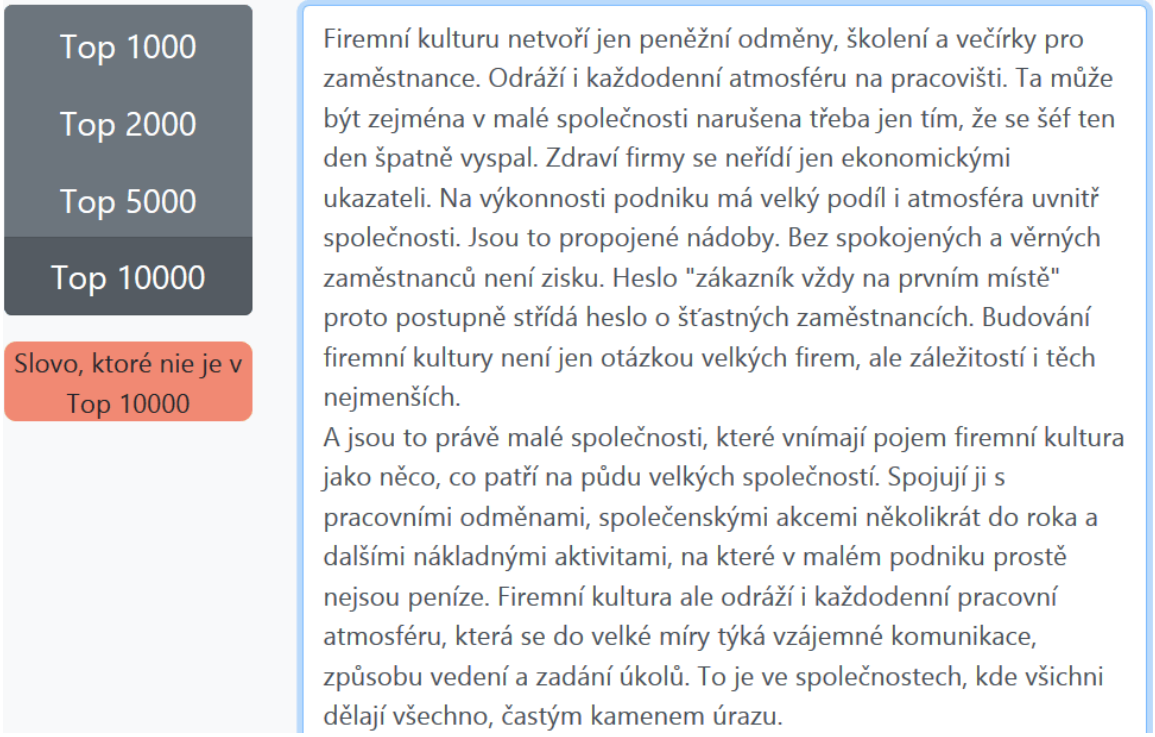
Príloha A

Výsledok aplikácie a niekoľko ukážok od užívateľov

A.1 Užívateľ č. 1



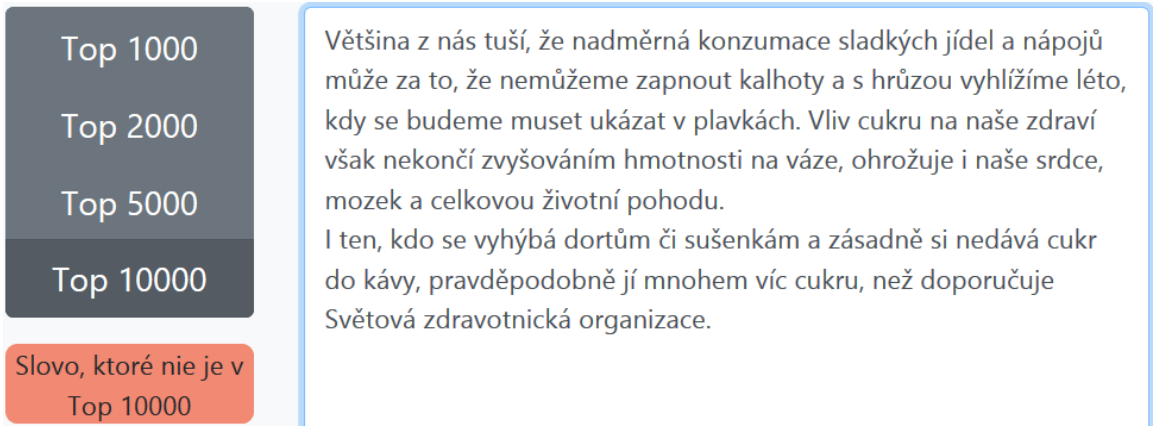
Obr. A.1: Obrázok ukazujúci riešenie aplikácie. Zobrazený pôvodný text užívateľa bez úprav.



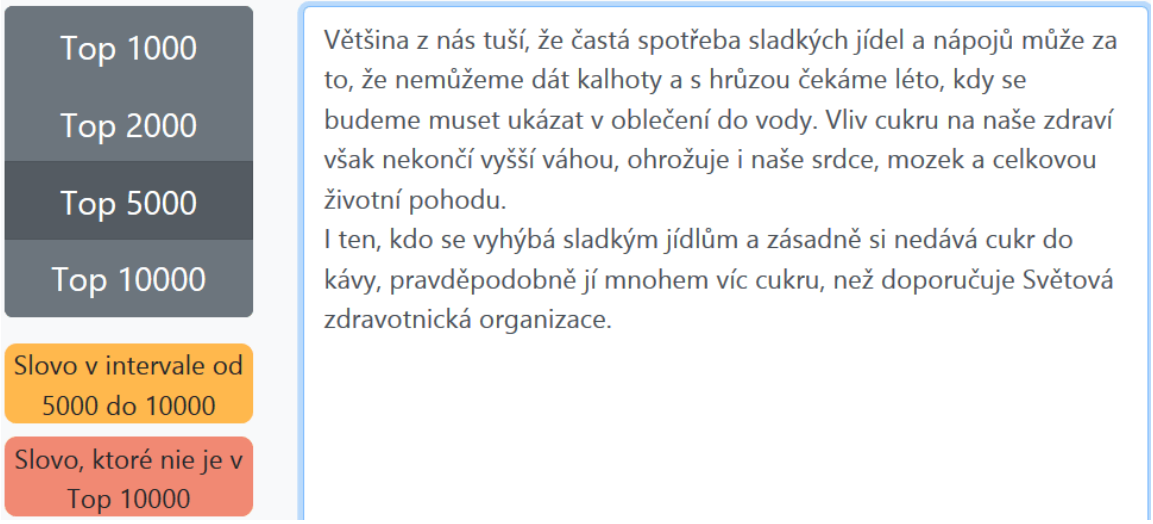
Obr. A.2: Užívateľovi sa podarilo upraviť text na najčastejších 10000 slov.

A.2 Užívateľ č. 2

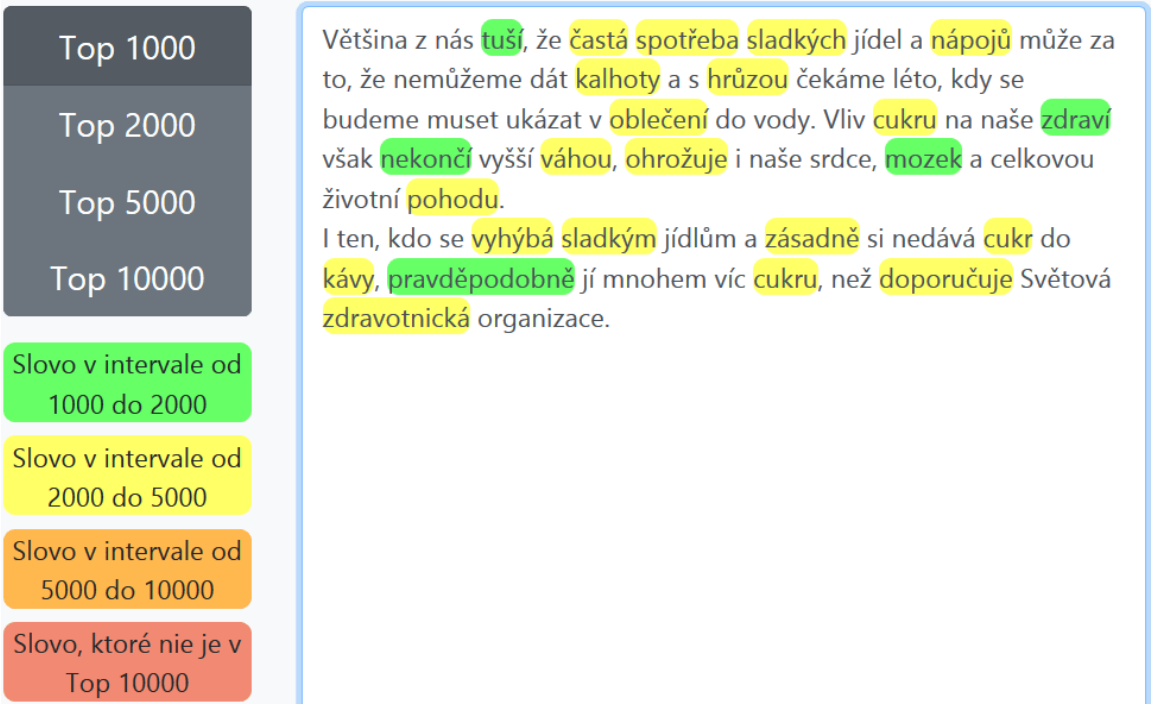
Obr. A.3: Obrázok ukazujúci riešenie aplikácie. Zobrazený pôvodný text užívateľa bez úprav.



Obr. A.4: Tento obrázok ukazuje upravený text s využitím top 10000 slov.

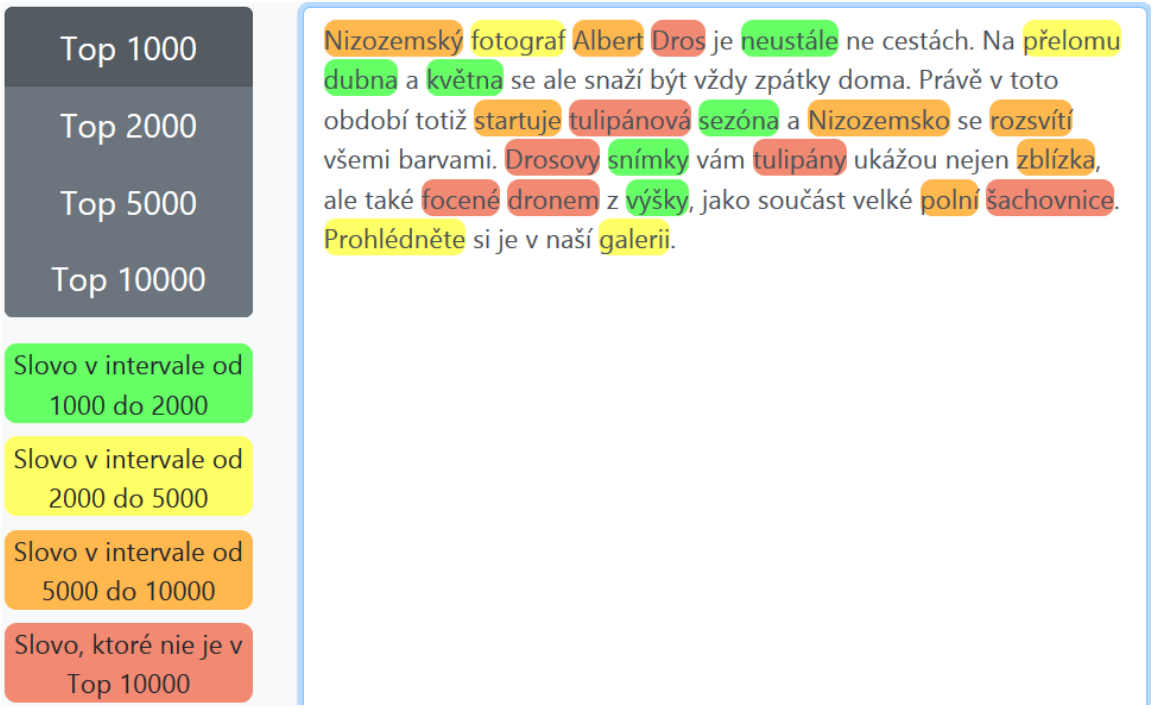


Obr. A.5: Užívateľovi sa podarilo upraviť pôvodný text pomocou top 5000 slov.

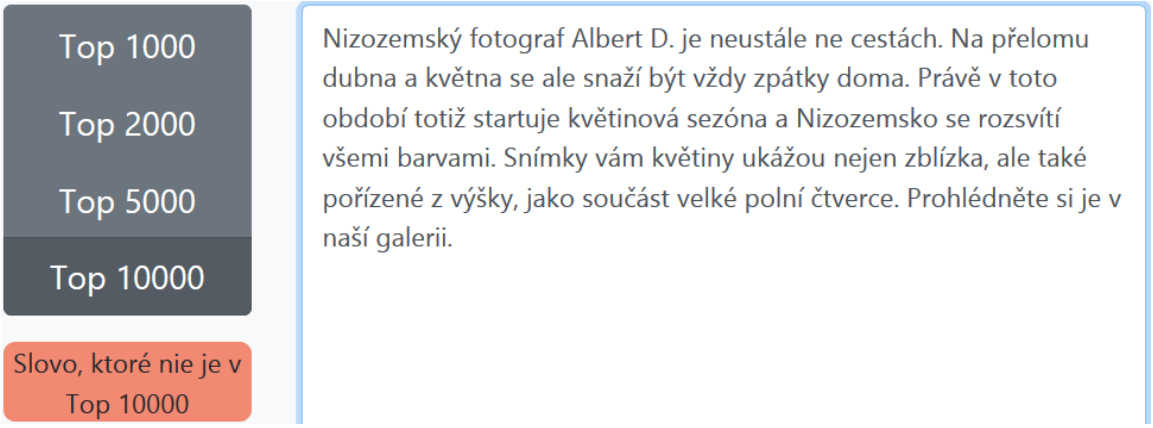


Obr. A.6: Na tomto obrázku je vidieť, ktoré časti textu už užívateľ nevedel upraviť.

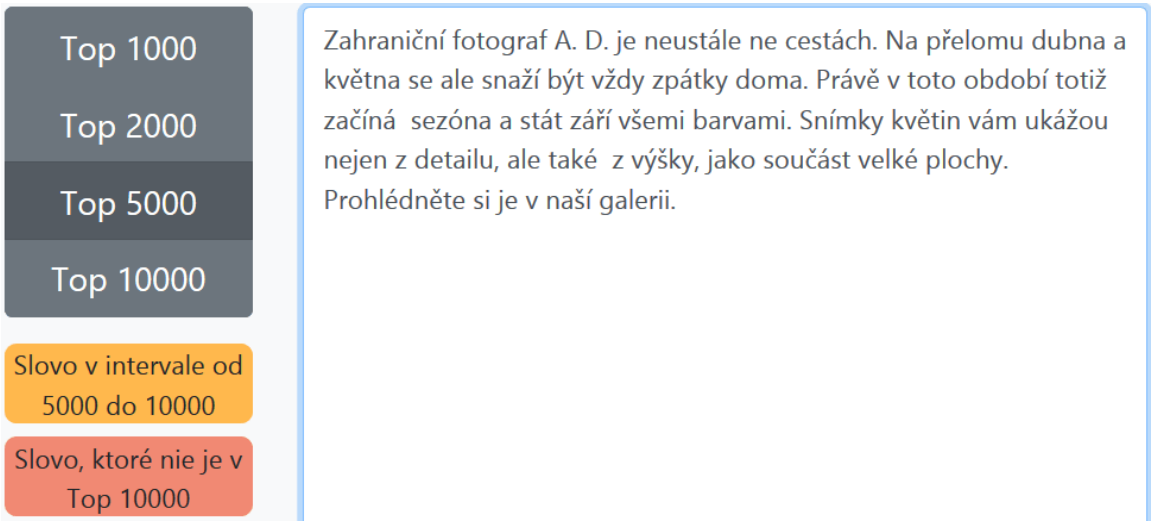
A.3 Užívateľ č. 3



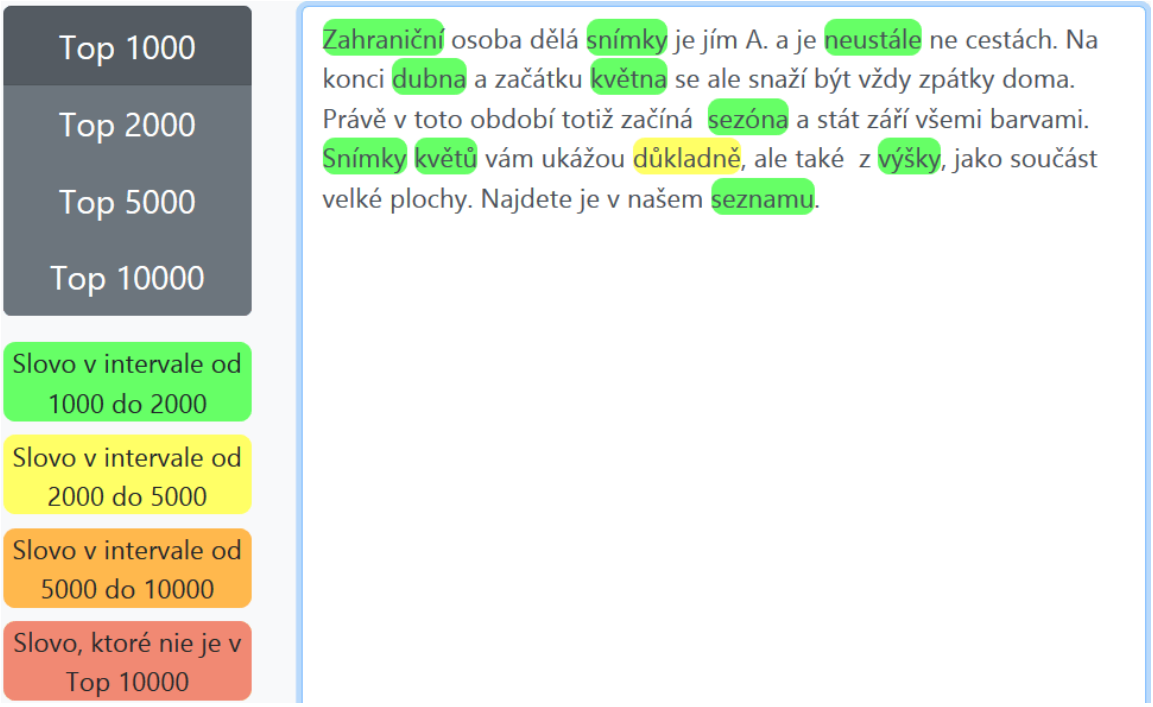
Obr. A.7: Obrázok ukazujúci riešenie aplikácie. Zobrazený pôvodný text užívateľa bez úprav.



Obr. A.8: Tento obrázok ukazuje upravený text s využitím top 10000 slov.



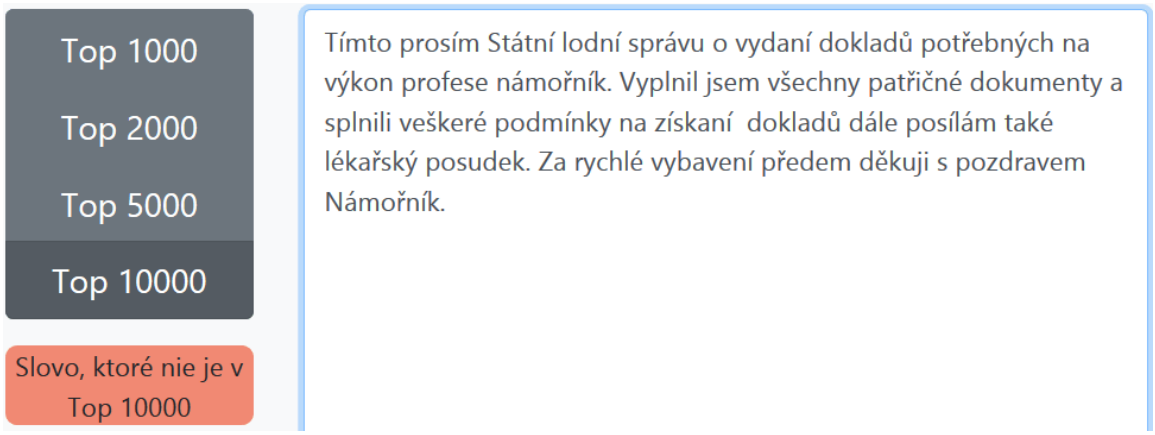
Obr. A.9: Tento obrázok ukazuje upravený text s využitím top 5000 slov.



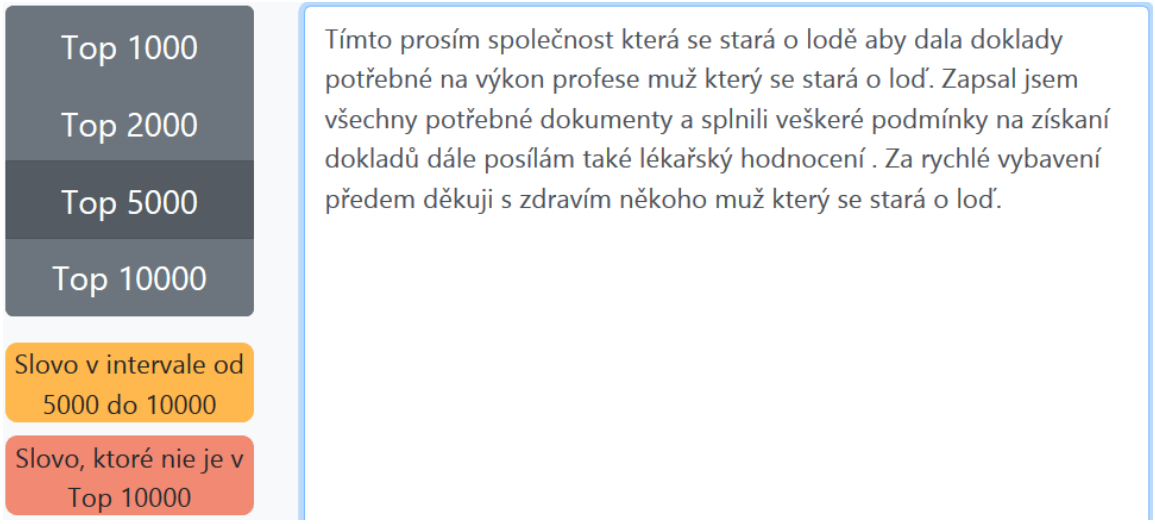
Obr. A.10: Užívateľovi sa nepodarilo upraviť pôvodný text na najčastejších 1000 slov, dokonca jedno slovo nie je ani z intervalu do 2000 najčastejších slov.

A.4 Užívateľ č. 4

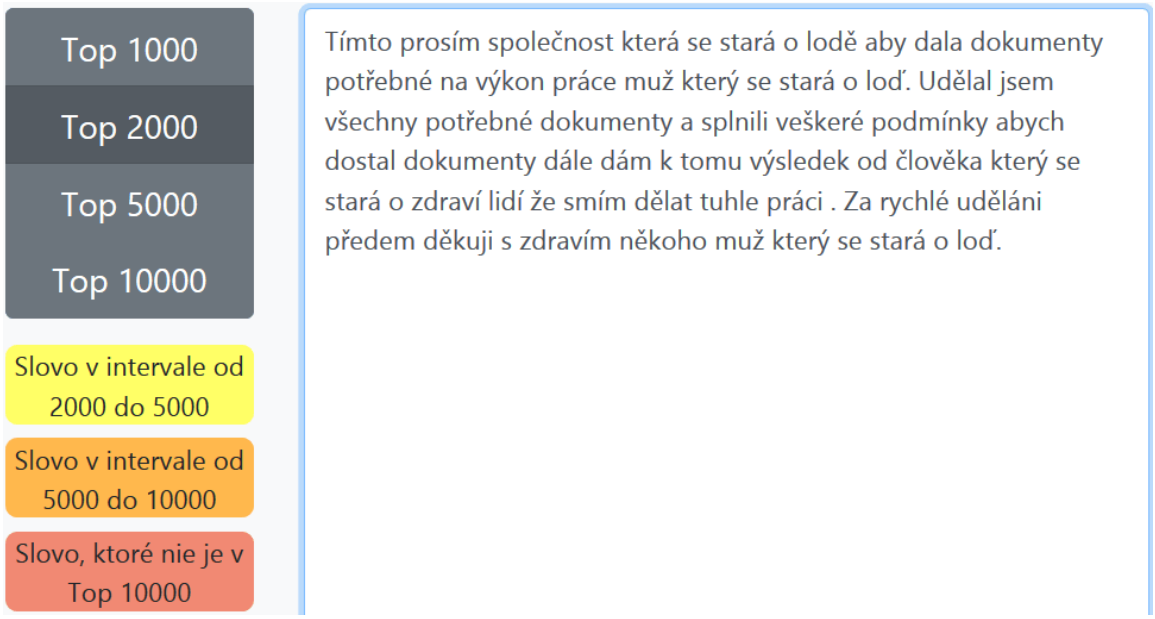
Obr. A.11: Obrázok ukazujúci riešenie aplikácie. Zobrazený pôvodný text užívateľa bez úprav.



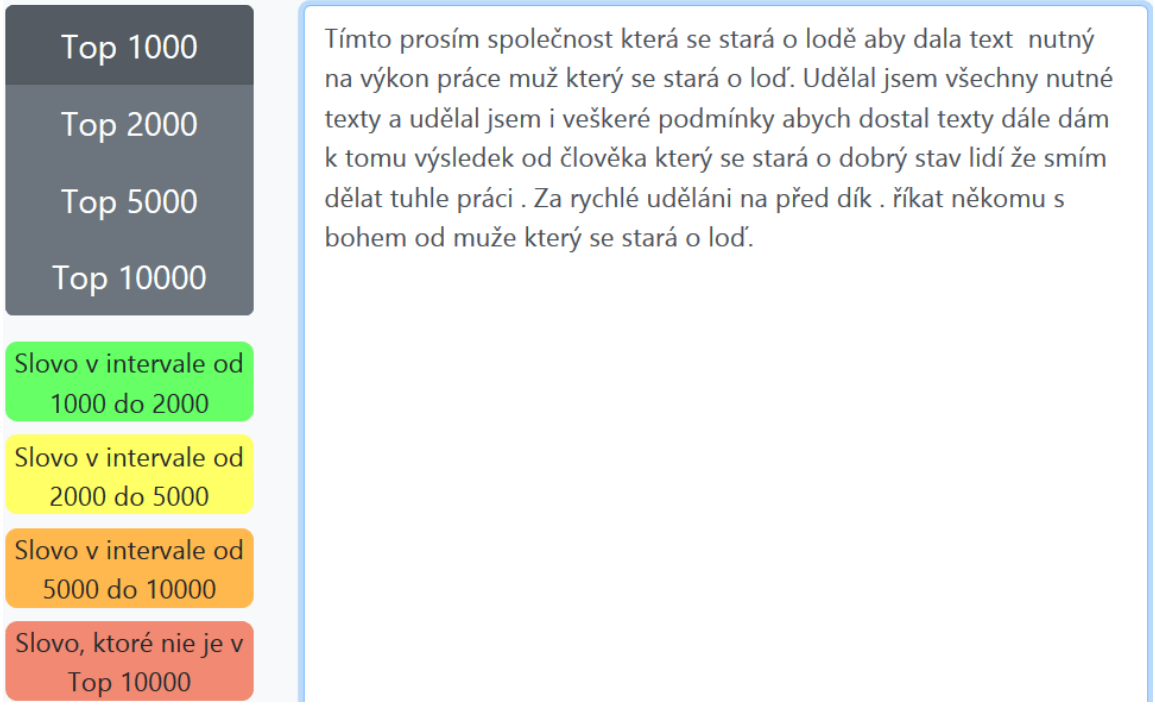
Obr. A.12: Tento obrázok ukazuje upravený text s využitím top 10000 slov.



Obr. A.13: Tento obrázok ukazuje upravený text s využitím top 5000 slov.



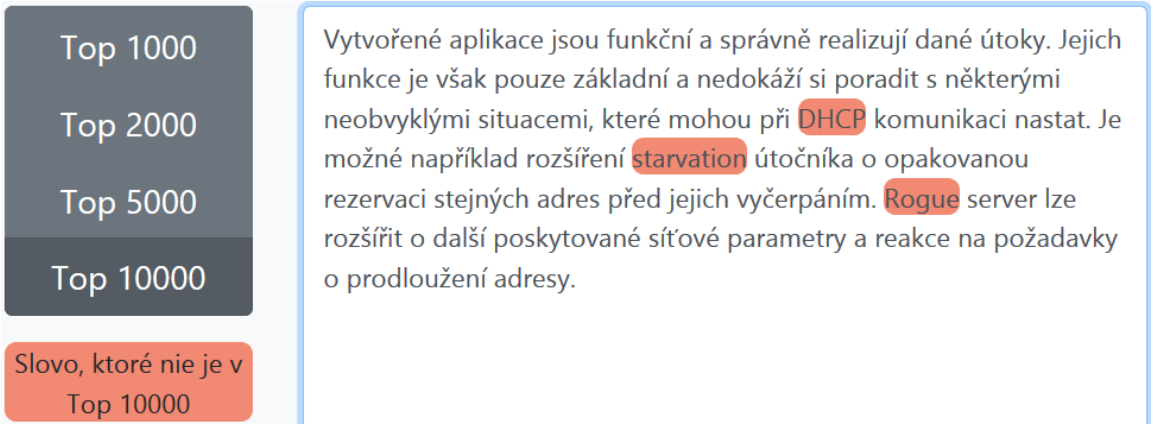
Obr. A.14: Tento obrázok ukazuje upravený text s využitím top 2000 slov.



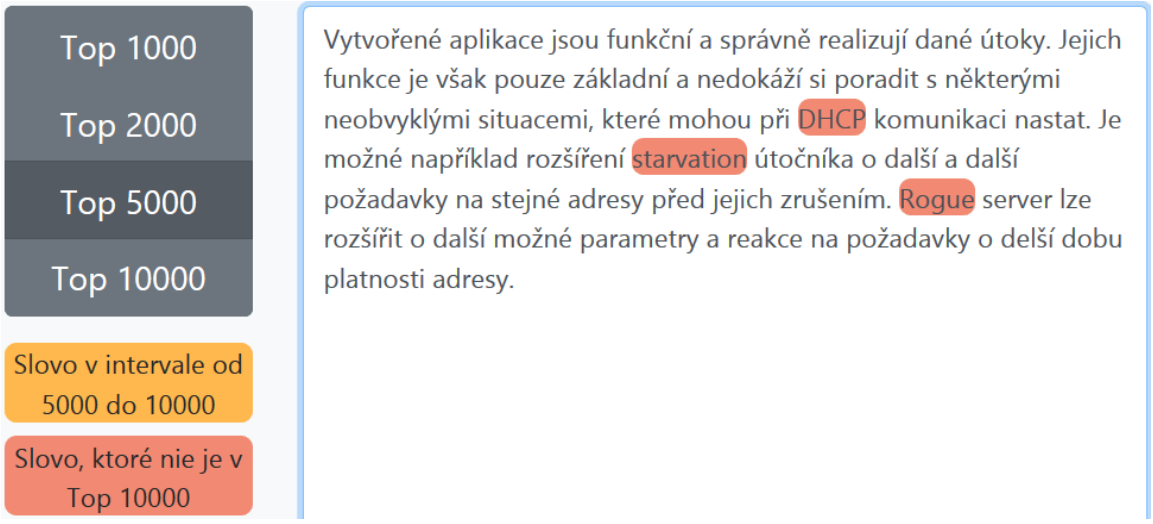
Obr. A.15: Užívateľovi sa podarilo upraviť pôvodný text s využitím len 1000 najčastejších slov.

A.5 Užívateľ č. 5

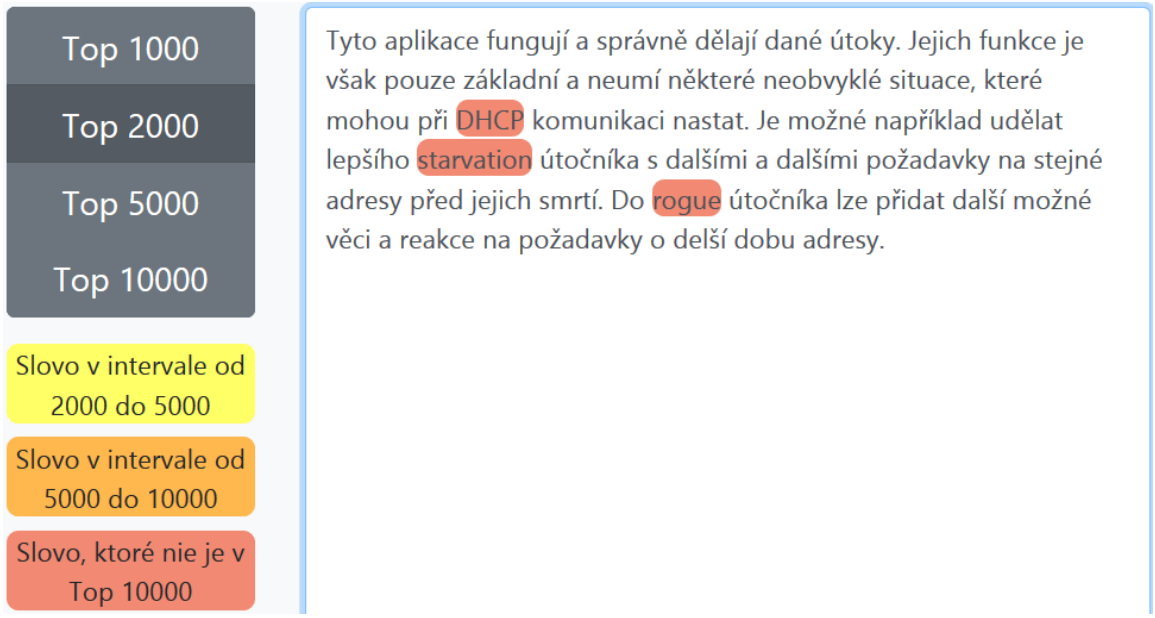
Obr. A.16: Obrázok ukazujúci riešenie aplikácie. Zobrazený pôvodný text užívateľa bez úprav.



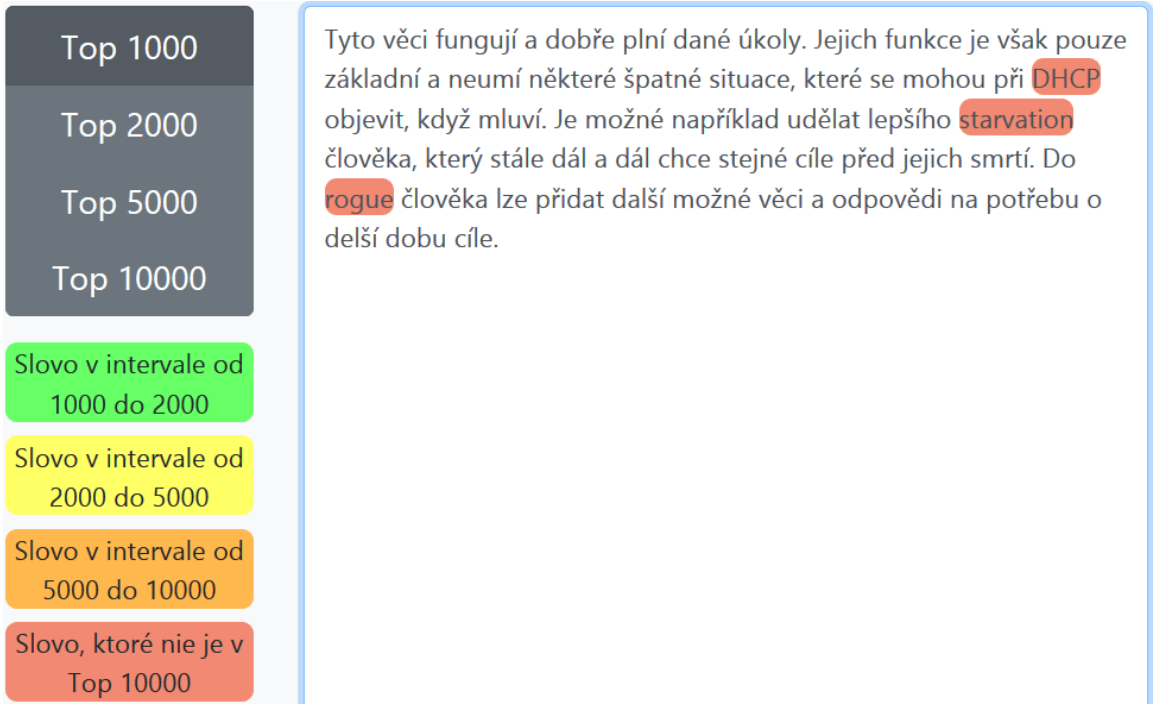
Obr. A.17: Tento obrázok ukazuje upravený text s využitím top 10000 slov. Užívateľ tu neupravoval slová, ktoré vyznačovali konkrétne pojmy.



Obr. A.18: Tento obrázok ukazuje upravený text s využitím top 5000 slov. Užívateľ neupravoval pojmy.

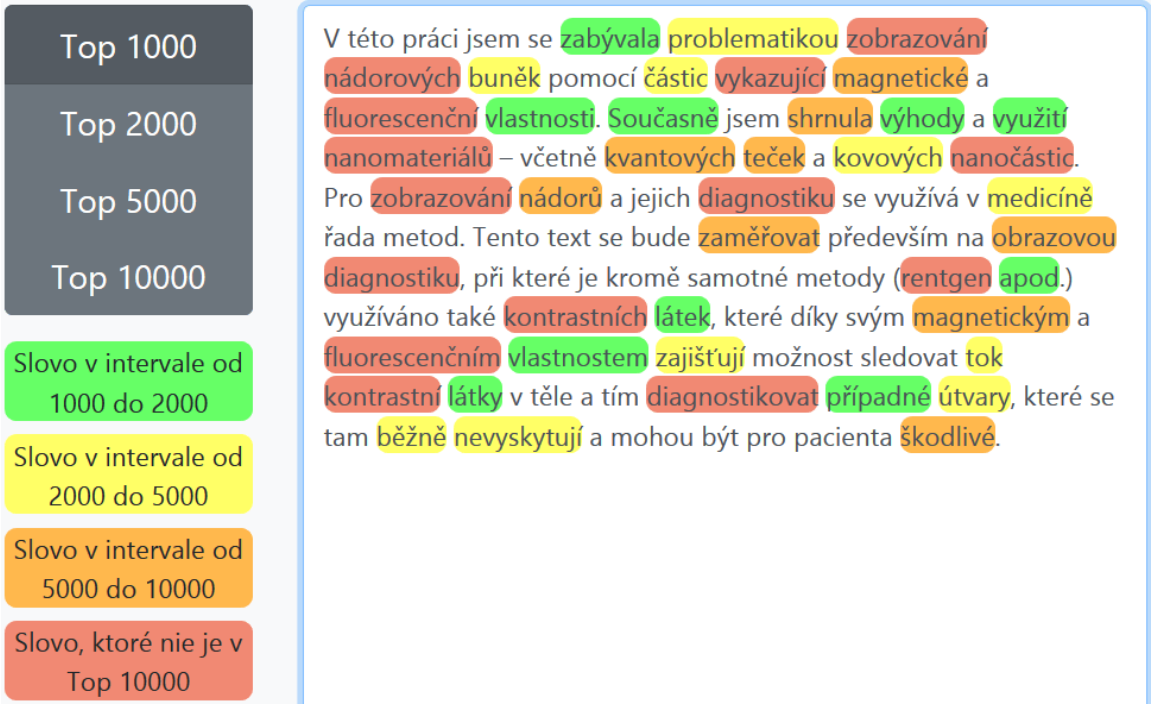


Obr. A.19: Tento obrázok ukazuje upravený text s využitím top 2000 slov. Užívateľ neupravoval pojmy.

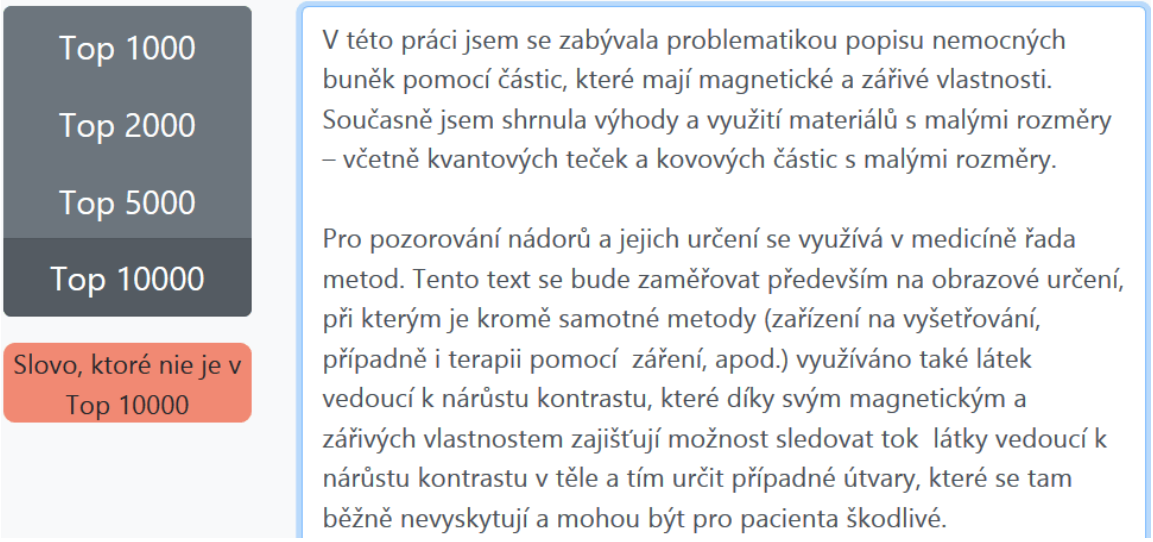


Obr. A.20: Užívateľovi sa podarilo upraviť pôvodný text s využitím len 1000 najčastejších slov. Užívateľ neupravoval pojmy.

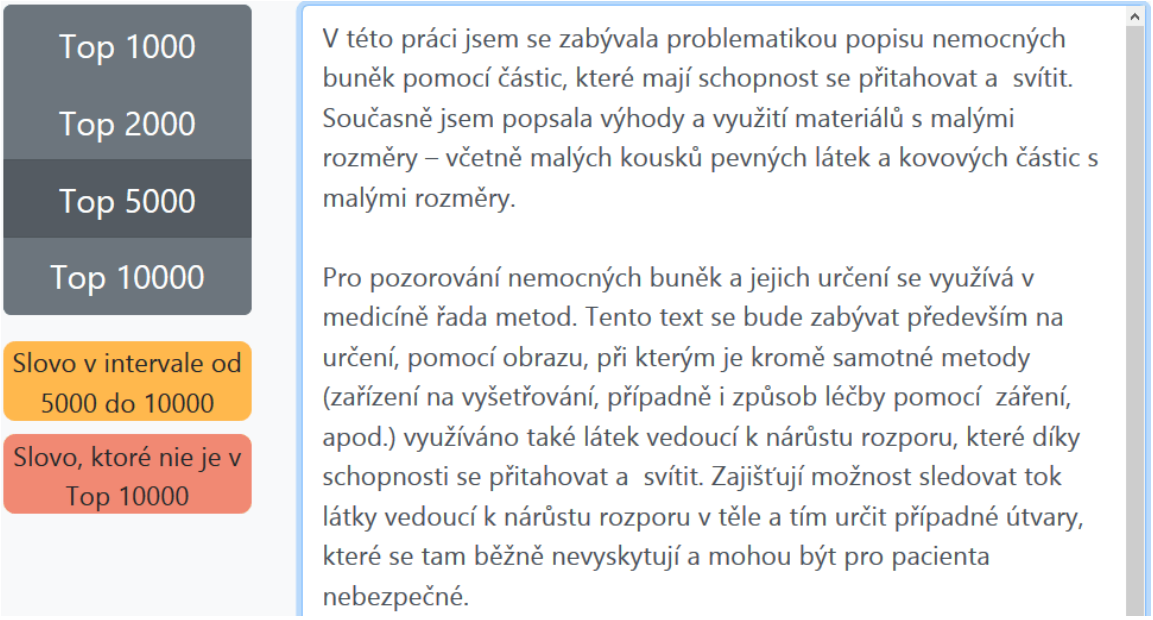
A.6 Užívateľ č. 6



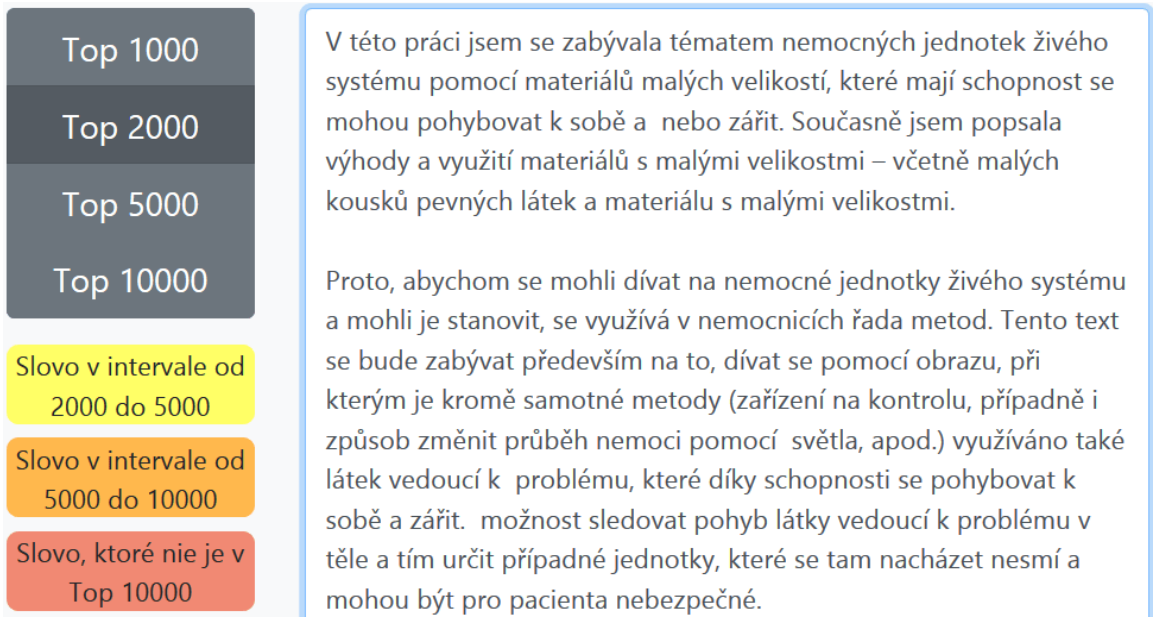
Obr. A.21: Obrázok ukazujúci riešenie aplikácie. Zobrazený pôvodný text užívateľa bez úprav.



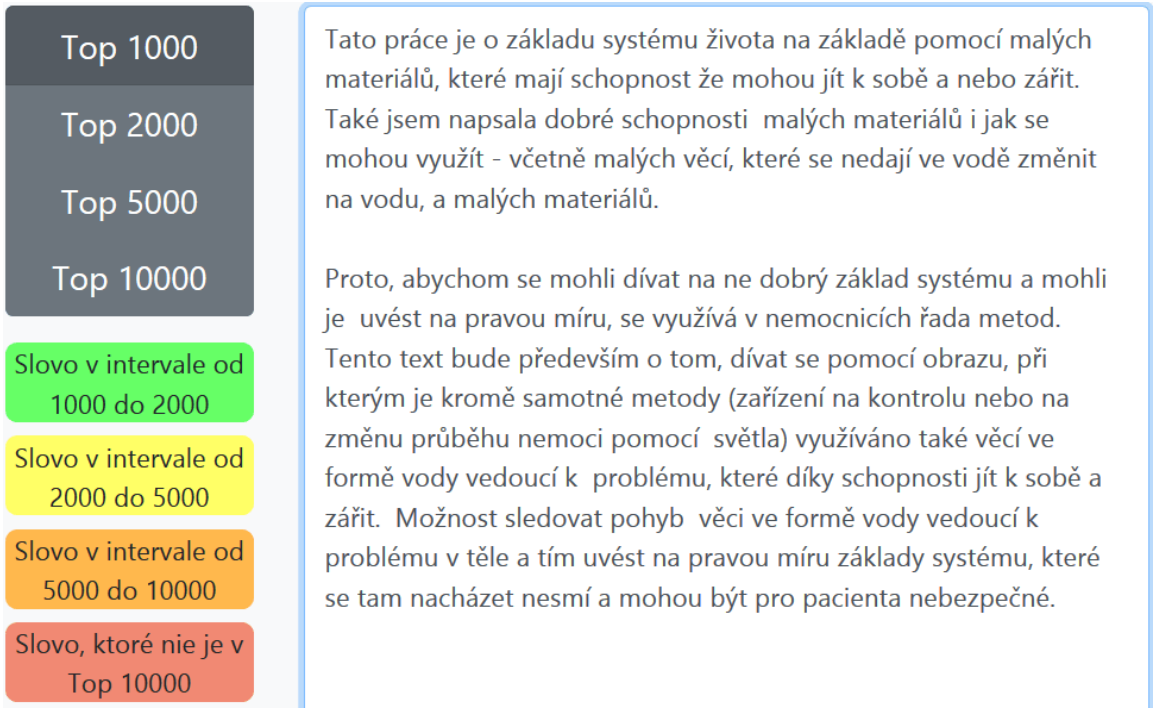
Obr. A.22: Tento obrázok ukazuje upravený text s využitím top 10000 slov.



Obr. A.23: Tento obrázok ukazuje upravený text s využitím top 5000 slov. Okrem toho sa na obrázku nachádza posuvná lišta v textovom poli, ktorá sa zobrazí len vtedy, ak je text dlhší ako textové pole.



Obr. A.24: Tento obrázok ukazuje upravený text s využitím top 2000 slov.



Obr. A.25: Užívateľovi sa podarilo upraviť pôvodný text s využitím len 1000 najčastejších slov.